AD-774 969

IMAGE PROCESSING SOFTWARE DOCUMENTATION

UNIVERSITY OF SOUTHERN CALIFORNIA

PREPARED FOR

ADVANCED RESEARCH PROJECTS AGENCY

DECEMBER 1973

UNCLASSIFIED
Security Classification

AD 774 969

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Image Processing Institute, Electronic Sciences Laboratory, University of Southern California, University Park, Los Angeles, California, 90007	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED
	2b. GROUP
3. REPORT TITLE IMAGE PROCESSING SOFTWARE DOCUMENTATION	
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) Technical Report	
5. AUTHOR(S) (First name, middle initial, last name) James M. Pepin, William K. Pratt, Guner Robinson, Mark Sanders	
6. REPORT DATE October 1973	7a. TOTAL NO. OF PAGES ~~199~~ 201 / 7b. NO. OF REFS
8a. CONTRACT OR GRANT NO. F08606-72-C-0008 NEW	9a. ORIGINATOR'S REPORT NUMBER(S) IPI Report 500
b. PROJECT NO. ARPA Order No. 1706	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) None
d.	
10. DISTRIBUTION STATEMENT Approved for release; distribution unlimited.	
11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209

13. ABSTRACT

This report contains descriptions of image processing software developed by the University of Southern California Image Processing Institute. Also included are operational descriptions of the IBM 360/44 and the HP-2100 computer systems ~~utilized~~ used in image processing.

****

14. Key Words: Image Processing, Digital Image Processing, Image Processing Software, Image Coding, Image Enhancement and Restoration, Image Data Extraction.

DD FORM 1 NOV 65 1473

UNCLASSIFIED
Security Classification

1

USCIPI Report 500

IMAGE PROCESSING SOFTWARE DOCUMENTATION

December 1973

William K. Pratt
James M. Pepin
Guner Robinson
Mark Sanders

Image Processing Institute
University of Southern California
University Park
Los Angeles, California 90007

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Eastern Test Range under Contract No. F08606-72-C-0008, ARPA Order No. 1706

ABSTRACT

This report contains descriptions of image processing software developed by the University of Southern California Image Processing Institute. Also included are operational descriptions of the IBM 360/44 and the HP-2100 computer systems utilized in image processing.

TABLE OF CONTENTS OF SECTIONS

I - ECL FACILITIES MANUAL

1. INTRODUCTION

The purpose of this Facilities Manual is to supply answers to the most common questions regarding the use of the computational facilities of the Engineering Computer Laboratory (ECL). The material will deal mainly with the use of the IBM 360/44. Included is a discussion on processing procedures, system notes, programming and debugging hints, operating schedule and priorities, system subroutines, and a list of error messages.

Almost all IBM manuals pertinent to our system are available in the Engineering Computer Laboratory. If you are interested in buying any of these manuals, please contact the lab manager.

Other available items which can be purchased through the lab manager are binders, indexes, flow charts, templates and coding forms.

2. PROCESSING PROCEDURES

A. How to Obtain a Job Number

All programs are run under job number control. A six-digit job number will be given to each user by the lab manager. Each user is responsible for the charges against his job number. Funded and unfunded job numbers are obtained as follows.

1. Funded users should obtain a funded job request form, fill it out completely, and return it to the business manager. Be sure to include the contract number and problem description. Funded users should also fill out a Purchase Request (PR) form and return it to the business manager. A sample PR is shown on the next page.

2. Unfunded users should obtain an unfunded job request form, fill it out completely, and obtain their sponsoring faculty's signature. Return the form to the business manager.

B. Charging Scheme for the E.C.L.

1. \$60.00 per hour for CPU time.
2. \$60.00 per CPU hour of core used. (This charge applies only to Foreground and regular Background jobs.) An hour of core is 100K.
3. Adage jobs are charged \$30.00 per elapsed hour of running time.
4. Each job requesting a private pack is charged \$0.25 per private pack.

   Examples:
   a. A Background job that uses 50K of core for 10 CPU minutes is charged as follows:
      10/60 x \$60.00 + 10/60 x 50/100 x \$60.00 = \$10.00 + \$5.00 = \$15.00
   b. A Background job of 5 minutes 100K and two private disk packs:
      5/60 x 60 + 6/50 x 100/100 x 60 + 2 x \$0.25 = \$10.50

c. A rolljob that runs 2 hours and requires no private packs:
120/60 x 60 = $120.00

d. A Foreground job that uses the Adage required 20K of core uses 10 CPU minutes and 4 elapsed time hours:
10/60 x 60 + 10/60 x 20/100 x 60 + 4 x $30.00 = $10.00 + $0.50 + $120.00 = $130.50

e. A Foreground job of 40K, no Adage and 30 CPU minutes and one private pack:
30/60 x 60 + 30/60 x 40/100 x 60 + $0.25 = $30.00 + $12.00 + $0.25 = $42.25.

DISK CHARGES

For storage of data on resident diskpack a charge of 5¢ per cylinder per day of allocation space. These are the two resident packs SSLRES & NEWSPL.

For storage of data on E.C.L. packs HOTDOG and USER01 a charge of 1¢ per cylinder per day is charged.

Example:
If account EEUSRB has 5 cylinders on SSLRES, 10 cylinders on NEWSPL and 20 cylinders on HOTDOG his daily charge of 95¢.

Some notes on what CPU time means in USC PS:

CPU time is all the time your program is dispatched. By dispatched we mean the time your program is using the CPU plus all wait state time charged to you. If a Foreground wants the CPU and you enter a wait state you are not charged for the wait state. But if you are alone in the machine you are charged for the wait time. Time spent in job control is not charged against your CPU time. So disk mount wait time is not added to your CPU time, but tape mount time is. In short, CPU time is the elapsed time your job would run if it had the 44 all to itself.

C. How to Prepare a Program and Run it as a Job

1. Write your program in 360 Fortran IV and/or 360 assembly language. IBM coding forms are available through the lab manager.

2. Keypunch your program on IBM cards using any of the 11 keypunches available for this purpose.

3. Obtain the necessary control cards from the card rack which is located in the keypunch room. See Figure 4 for necessary control cards for each specific case.

4. Punch your last name (up to 8 characters) starting in column 3 and your six-digit job number starting in column 17 of the job card.

5. Prepare your job deck according to Figure 1, as applicable.

6. Place your deck in the appropriate box located in the computer room. (If unable to decide which box to put it in, as the operator for assistance.)

7. Your job will be run by the operator on a first-come-first-serve basis, and will be placed with printout in the output box (it may be handed to you directly if job traffic is light).

8. Remove your job from the output box. If corrections or modifications to your program are necessary, please make the changes in the user area before returning for another run.

D. Deck Set Up

In this section several examples of setting up a Fortran job deck are illustrated.

1. Simple deck without subroutines: Set up your job deck with control cards using the following order of cards specified (see Fig. 1A):

a. White "JOB" control card with your name and job number.
b. Red control card (// EXEC FORTRAN).
c. Your punched program deck, referred to as "main", without data cards.
d. Yellow control card (// EXEC LOADER).
e. Your punched data cards.
f. Blue control card (/& END OF JOB).

2. Deck with three-Fortran subroutines: In this case the following order is recommended (see Fig. 1B:).

White "JOB" control card with your name and job number.

Red control card (EXEC FORTRAN).

Your punched program deck.

One card of the form (SUBROUTINExxxxxx), where xxxxxx is the name of the subroutine.

Body of subroutine xxxxxx.

A (RETURN) card (this card can be anywhere in the subroutine but must be used in such a manner as to act as the executable card of the subroutine).

SUBROUTINE yyyyy.

Body of subroutine yyyyyy.

RETURN

USC PURCHASE REQUEST NO. ____________

DEPARTMENT Electrical Engineering

REQUISITION NO. ____________ OR

D.P.O. NO. ____________ P.O. NO. ____________

ACCOUNT NO. 10-1000-0000

DELIVER TO PERSON: Dr. Jones

BUILDING: OHE ROOM NO. 000

DATE NEEDED Jan. 1, 1975 AUTHORIZED BY ____________

ITEM NO.	CATALOGUE NO.	QUANTITY	DESCRIPTION	UNIT PRICE	TOTAL COST
CONFIRMED PURCHASE ORDER NUMBER ____________ ( )					
1			5 Hours of IBM 360 Time @$1.20/Hr.		600.00
2			1 Hour of AGT-10 Time @$30/hr		30.00 ******
			TOTAL		$630.00
			Vendor: Engr. Computer Lab.		
			PHE-ROOM 106		
			Attn. Business Mgr.		

VENDOR INFORMATION:
NAME ____________

ADDRESS ____________

CITY ____________ STATE ____________ PHONE ____________

ATTENTION OF ____________

END

SUBROUTINE zzzzzz.

Body of subroutine zzzzzz.

RETURN

END

Yellow control card (EXEC LOADER).

Your punched data cards.

Blue control card (/& END OF JOB).

A. JOB CARDS

The job card gives the system information about the nature and resources required by your job. This information includes such things as accounting information and resource requirements. In this section the position and format of these fields will be laid out.

The job cards has 3 fields in the first half of the job card. The name field, the account field and the comment field. On the white pre-punched job cards you put the name field starting in Col. 3 and maybe up to 8 characters in length. The account code begins in Col. 17 and is the 6 digit job number assigned to you when you applied for an account. The comment field is immediately following the comma after the account field. This field will come out on your print out, along with your job name, in block letters. In columns 73-80 there is provisions for more parameters concerning estimated execution time and resource requirements they are summarized here:

Col. 80 estimated CPU time

blank	1 min.
0	30 sec.
1	2 min (funded)
2	2 min (unfunded)
3	4 min (funded)
4	4 min (unfunded)
5	7 min (funded)
6	7 min (unfunded)
7	10 min (funded)
8	10 min (unfunded)
9	unlimited (note these jobs are held til non-prime time execution)

Col. 79 core required by your job

	bytes	hicore
blank	52k	1100016
0	56k	1200016
1	64k	1400016

2	72k	1600016
3	80k	1800016
4	88k	1A00016
5	96k	1C00016

Col. 78 system resources required

blank	ro adage, no foreground
1	adage,no foreground
2	no adage, foreground
3	adage, foreground
4	roll job

Col. 77 tape drives required

1	1-tape
2	2 tapes or the dual density drive

Col. 76 private disk drives

1	one private pack
2	two private packs

Col. 75 Holdprint option

H	Do not let job print

3. Fortran deck with assembly language subroutine: In this case the following order of cards should be used (see Fig. 1C):

a. White "JOB" control card containing your name and job number.
b. Red control card (EXEC FORTRAN).
c. Your punched program deck (main).
d. The card (// SUB EXEC ASSEMBLE)
e. Body of assembly language subroutine.
f. END card.
g. Yellow control card (// EXEC LOADER).
h. Your punched data cards.
i. Blue control card (/& END OF JOB).

For extensive and explicit explanation of the use of assembly language subroutines with Fortran programs, see Appendix C of "Guide to System Use for Fortran Programmers." This manual can be obtained from the computer operators

Note

Note that the name field[1] for the EXEC ASSEMBLE card must differ from the EXEC FORTRAN card. Fortran and assembly subroutines can be used simultaneously within the same program using similar formats as given in Figure 1B and 1C.

[1] The name field consists of columns 3 through 8

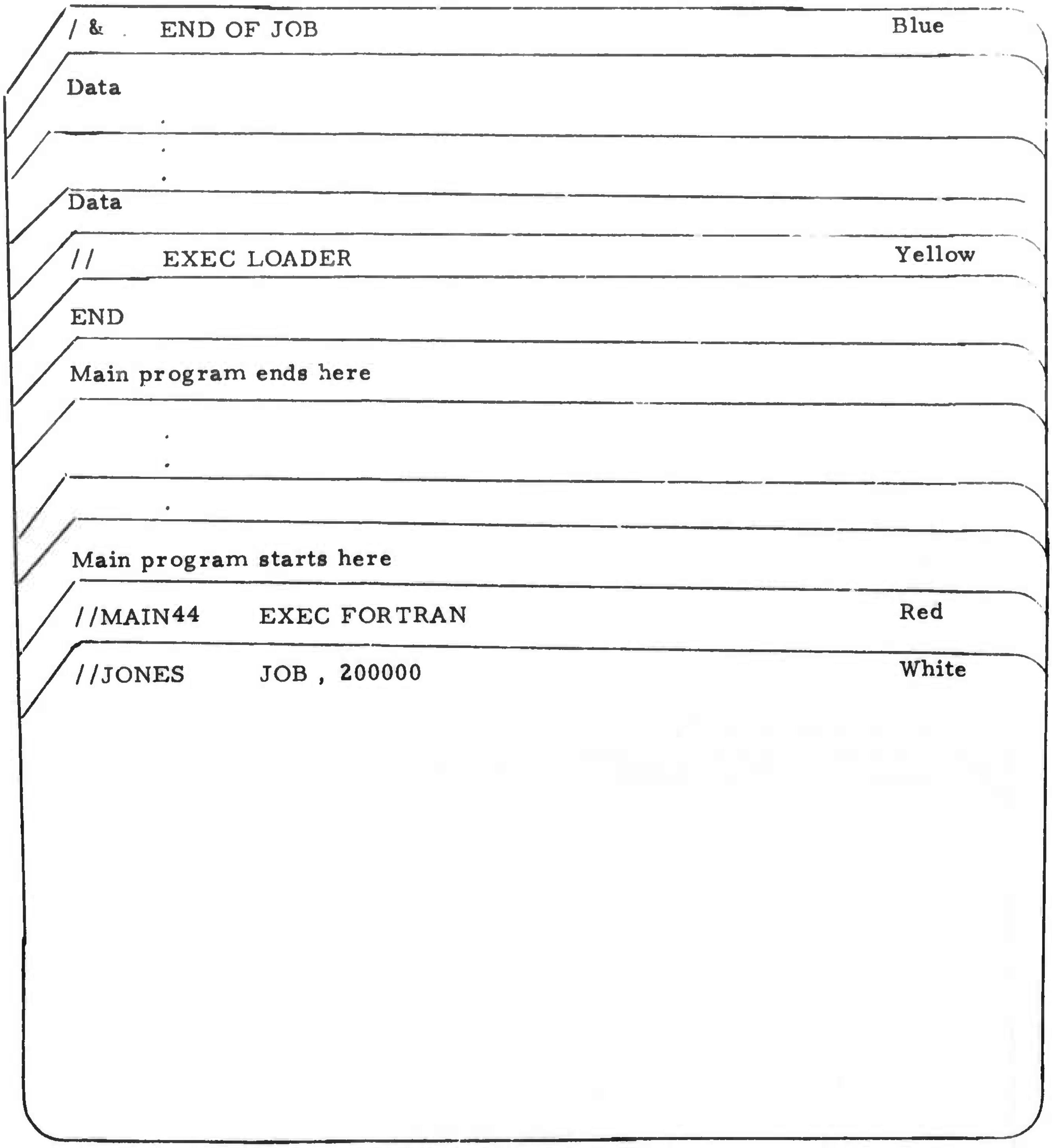
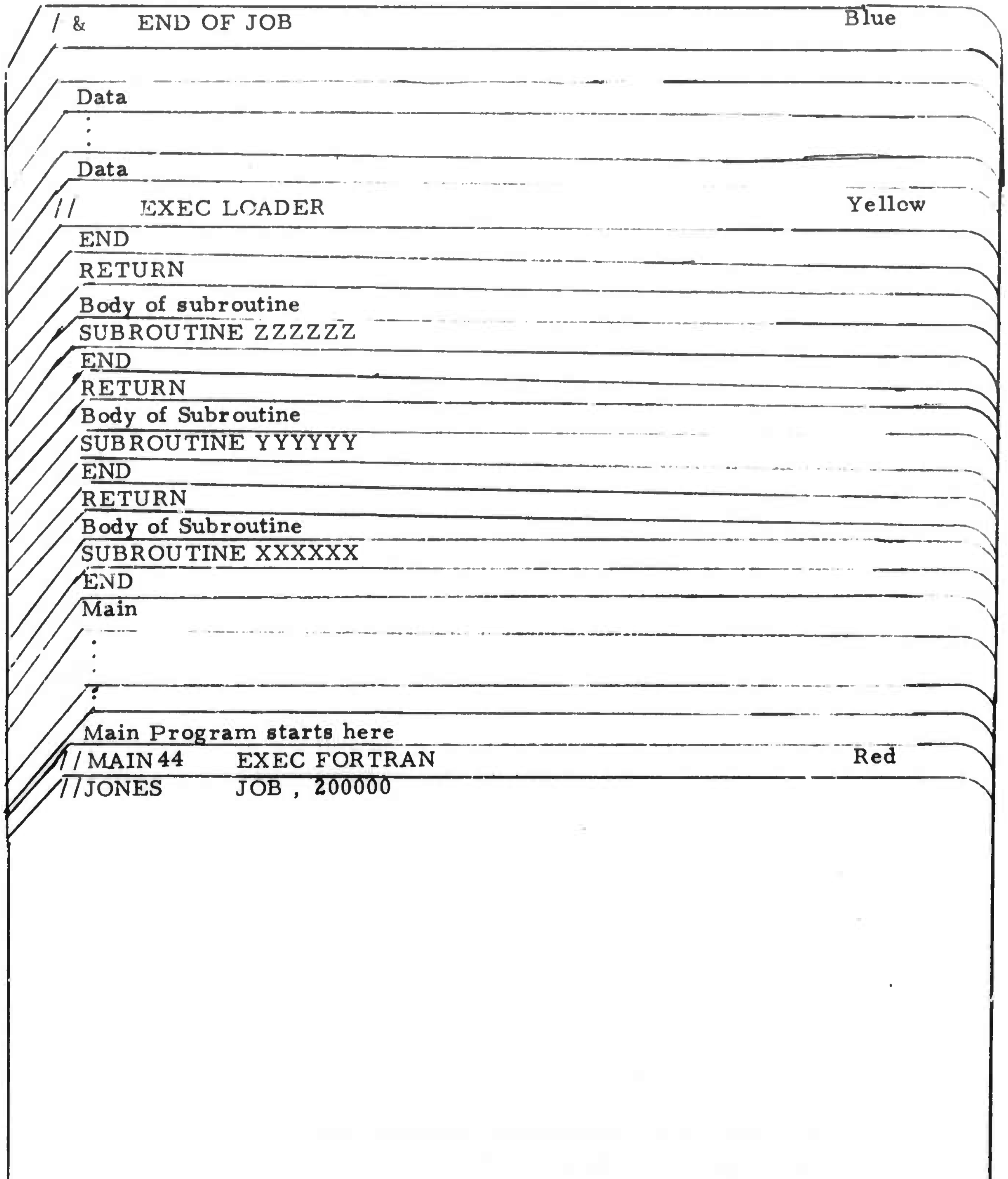
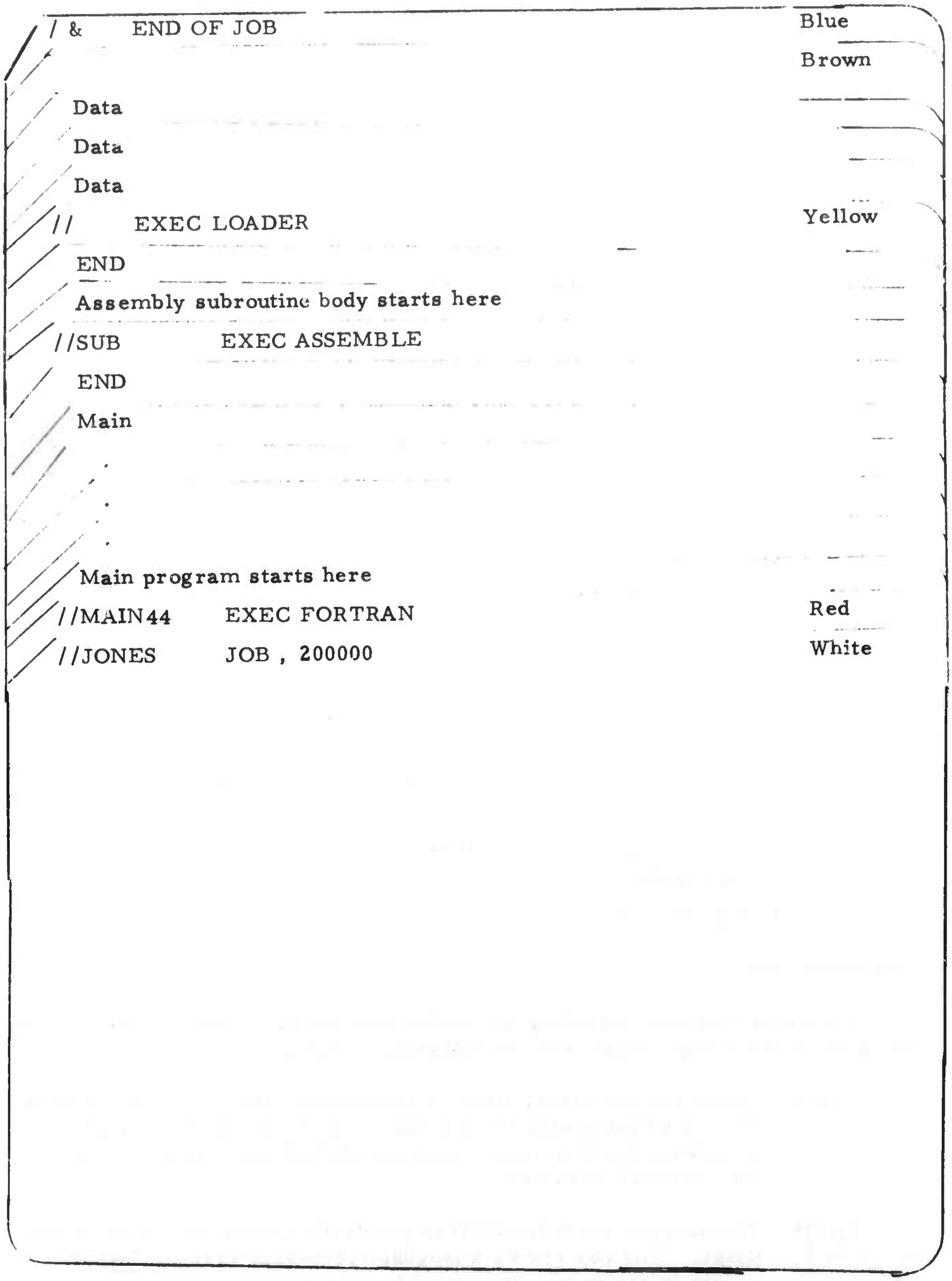


Figure 1A. Deck Set Up Without Subroutines.



```
/ &       END OF JOB                                  Blue
Data
 :
Data
//        EXEC LOADER                                 Yellow
END
RETURN
Body of subroutine
SUBROUTINE ZZZZZZ
END
RETURN
Body of Subroutine
SUBROUTINE YYYYYY
END
RETURN
Body of Subroutine
SUBROUTINE XXXXXX
END
Main
 :
Main Program starts here
//MAIN44     EXEC FORTRAN                             Red
//JONES      JOB , 200000
```

Figure 1B. Deck Set Up with Fortran Subroutines



```
/ &       END OF JOB

  Data
  Data
  Data
//        EXEC LOADER
  END
  Assembly subroutine body starts here
//SUB         EXEC ASSEMBLE
  END
  Main
    .
    .
    .
  Main program starts here
//MAIN44      EXEC FORTRAN
//JONES       JOB , 200000
```

Figure.1C. FORTRAN Deck with Assembly Language Subroutine

EXAMPLES

In all of the following examples, references to standard routines residing in the system library SDSREL will automatically be resolved.

1. To load and execute several compiled routines:

```
// EXEC LOADER
   data cards
/& END OF JOB
```

2. To load and execute a routine just compiled that calls a special subroutine from the user library SDSLIB:

```
//SYS003 ACCESS SDSLIB
//  EXEC LOADER(SYS003)
    data cards
/& END OF JOB
```

3. To load and execute a main program MAIN residing in the directoried data set SDSLIB:

```
//SYS000 ACCESS SDSLIB(MAIN)
//  EXEC LOADER
    data cards
/& END OF JOB
```

4. Execute a compiled program and request a core dump if the job is cancelec:

```
//  EXEC LOADER(MAP),DUMP[1]
    data cards
/& END OF JOB
```

DIAGNOSTICS

The error messages listed by the loader have the same number and meaning as those of the linkage editor with the following exceptions:

KA611 Phase size too large, there is insufficient core to load the program. Since the loader uses the top 360 bytes of core it cannot load programs into this area. However IBCOM may use this area for its run time I/O buffer.

KA721 The program contains address constants outside the limits of the phase. This represents a language processor error unless it occurs in conjunction with KA 611, insufficient core.

[1] This parameter is interpreted by the SSL accounting routine.

KA871 An invalid end of extent was detected while reading the directories of SYSREL or a user specified private library.

KA931 The system was unable to open SYS000, SYS001, or a user specified private library.

ADVANTAGES AND LIMITATIONS

1. Advantages

1. Eliminates LNKEDT and EXEC control cards.
2. Eliminates fetching job control twice.
3. Quick phase load: LOADER is less than 2000 words long.
4. Immediately cancels job with dumping suppressed if compiler or assembler had errors other than warnings (severity 4).
5. Doesn't use module or phase cards.
6. Ignores SYSPSD. Thus, "SYS000 tapes" can be saved and reused (e. g. instead of punching object decks).
7. Doesn't write on SDSABS.
8. Uses an optimum library search algorithm.
9. Minimum use of "RLD file" (SYS001).
10. Suppresses form ejects and list of created cards if not mapping.
11. Prints unresolved adcons even if map not requested.
12. Cancels job with dumping suppressed if any errors during loading.
13. Sets switches for short dump[2] (starting after IBCOM and ending at end of common) and a save area trace if job canceled.
14. Uses previously link edited IBCOM, FIOCS, UNITAB, USEROPT phase that is fetched only if no error in loading main program.

2. Limitations

1. System units cannot be reaccessed before execution. Therefore the user should plan to use SYS000 and SYS001 as scratch only, and have special data on other units.

2. Cannot build phases (user can fetch previously link edited phases).
3. Since INCLUDE cards for SYS000 inputs are not printed, errors in cards on SDS000 cannot be pinned down to specific modules.
4. Always loads IBCOM, the Fortran I/O subroutine.
5. Transfer address fixed as that of first one received on an end card. The order modules are read in SYSIPT, SYSOOO, SYSREL, and then user specified libraries. Assembly programmers should not specify a transfer on an end card if a routine is not a main program.
6. Has a smaller symbol table than the linkage editor.
7. Rep cards that reference locations also affected by RLD cards may be processed incorrectly. Because of the way RLD cards are handled (see 9 above), it is impossible to predict whether adcons will be added before (incorrect) or after (correct) REP information is inserted. Note that the majority of REP cards do not refer to locations affected by RLD cards.

---

[2] Dump will be taken only if user requested it.

Input to the loader consists solely of relocatable modules to be loaded. The possible sources of loader input are:

1. SYSIPT, the system input unit.
2. SYS000, usually the system unit written in a previous job step by the compiler or assembler.
3. SYSREL, the system library.
4. SYS002 - SYS009, private user libraries.

LOADER CONTROL STATEMENT

The loader is called by specifying its name on an EXEC statement:

```
// EXEC LOADER
```

Enclosed in parenthesis after the word LOADER can be any combination of the following four options (parameters):

MAP, MODULE, NOLINK, SYSxxx or SYSFxx

MAP

The parameter MAP requests that the loader produce on SYSLST a listing of the locations and names of all programs loaded.

MODULE

The parameter MODULE specifies that the user is including, in the input stream behind the EXEC LOADER statement, object decks to be loaded into core. The end of these decks must be denoted by a card with a slash in column 1. Notice that if this parameter is not given, a /* should not be put between the EXEC LOADER card and data cards for the run. Non-loader cards (no 12-9-2 punch in column one) are listed on SYSLST and ignored. Thus link editor control cards or member header cards can be left in decks.

NOLINK

The parameter NOLINK specifies that the loader is not to read the intermediate data set SDS000 for load modules. This option is required for jobs without previous compilation or assembly steps.

SYSxxx

These parameters specify the directoried user libraries which are to be searched for subprograms called by other modules. There can be up to four of these parameters, where xxx is 002, 003, 004, or 005, and SYSxxx is a unit previously accessed to a directoried data set.

3. SYSTEM NOTES AND GUIDELINES

A. Linkage Editors

To convert assembler and computer output modules (i. e., Fortran and assembly language programs) into a form suitable for loading and execution, it is necessary to use the system's linkage editor program. All programs to be executed under system control must first be processed by the linkage editor, Alvadu. In the following section the linkage editor's options and explicit explanation of each option are discussed.

Two linkage editors are provided for linking user programs. They are named LOADER and LNKEDT. Since it is considerably faster than the others, the first editor (LOADER) should be used for all programs which do not consist of multiple phases. This is accomplished via the EXEC LOADER card. (If you do not know what multiple phases are, then your program does not consist of multiple phases). The use of LOADER is described on page 8 of this manual. The second editor (LINKEDT) should be used for linking multiple phase programs. This version allows usage of a user library in the same manner as LOADER. Otherwise, the parameters of LNKEDT are identical to those of the standard LNKEDT, which are described in several IBM publications.

B. 44 PS Loader (LOADER)

This is a routine designed to replace the link-edit and to execute steps of normal Fortran batch job processing. It provides for quick program loading while not sacrificing the multi-library facility of RLNKEDT.

LOADER

Output from the compiler or assembler is in the form of relocatable object program modules. Each module consists of an external symbol dictionary, the text of the module, and a relocation distionary. These modules must be converted to absolute form and loaded into core before program execution can begin.

The loader provides facilities for moving relocatable text into core, replacing address constants by their absolute values, resolving symbolic cross-references between modules, and transferring control to the loader program.

LOADER PROCESSING

The programmer may specify at compilation or assembly time which modules are to be produced for the loader. In addition, he can specify private libraries from which the loader is to automatically retrieve referenced modules. Finally, he can include module decks punched in previous jobs (in the input stream for the loader).

SYSTEMS UNITS

SYSIPT	Input modules (I) if MODULE option used.
SYSLST	Map and error messages
SYS000	Input modules (II) unless NOLINK option used.
SYS001	RLD work file.
SYSPSD	Ignored.
SYSREL	Autolinked modules (III).
SYSxxx	Autolinked modules (IV) if SYSxxx option used. xxx may be 002, 003, 004, 005, 006, 007, 008, 009.

C. Fortran Reference Numbers and Device Assignments

Input/Output statements in Fortran require specification of a data set reference number. Each reference number has a fixed assignment to a symbol unit (this unit is not under programmer's control), and each symbol unit has a standard assignment to system data set (this assignment can be modified by ACCESS control card).

The standard assignments are:

Fortran Unit #	Symbolic Unit	Data Set
1	SYSF01 or SYS001	SDS001
2	SYSF02 or SYS002	U. A. (unassigned)
3	SYSF03 or SYS005	U. A.
4	SYSF04 or SYS004	U. A.
5	SYSF05 or SYSIPT	Card Reader
6	SYSF06 or SYSOPT	Printer
7	SYSF07 or SYSPCH	SDSPCH
8	SYSF08 or SYS000	SDS000
9	SYSF09 or SYS006	U. A.
10	SYSF10 or SYS007	U. A.
11	SYSF11 or SYS008	U. A,
12	SYSF12 or SYS009	U. A.

Usage of units other than 1-12 is illegal. The data sets SDS000, SDS001 and SDSPCH are available as temporary storage areas. For further information see the IBM manuals on how to DISK I/O in Fortran.

D. Foreground Execution

The Operating System in the 360-44 has a means for execution of several foreground jobs. This process is done by the movement of the subject job to a new region of core in the high part of the 360 core. This movement allows the cpu to move on to the next batch job in the input stream so as to overlap the execution of the foreground jobs with the background batch stream. The foreground is useful for many applications among them are I/O bound jobs, graphics jobs and special data acquisition jobs. To run a foreground job you must run a special processor like FORTRAN or the LNKEDT program its name is SSLOADER and it is described next.

E. SSLOADER

Is designed to be a general usage call to CATTAC, the foreground program attacher, to load programs in a multiprogramming partition. It is intended for ADAGE and any other program that will operate best in a temporary partition.

Before one can execute SSLOADER the user must compile his program and then execute LNKEDT. The word RELO must be given as an option to the linkage editor. It signifies that your program must be made relocatable so that it can be loaded anywhere in core to be executed. This is needed because the system will not have a fixed location for the foreground it can move from run to run. For most FORTRAN programs it will be sufficient to just have // EXEC LNKEDT(RELO). However, there will be some programs that because of size or other reasons are more than one phase programs. Any program that is larger than one phase and as a result does a LOAD or FETCH, has to specify KEEP as an argument when the linkage editor is executed. If this is not done after the user's program is into a foreground partition and the next user comes into the background the previous user's phases will be deleted. After the linkage editor has been executed, the programmer should now execute SSLOADER to put his program into a foreground partition. SSLOADER is so designed that most programs will have to merely execute it. However, some programs will want and need different options and they can specify nine different options. The options are listed on the cards following the // EXEC SSLOADER card. If more than one option is listed per card they must be separated by a comma. A comma should not follow the last option on a card. The order of the options make no difference in the general case and if it does matter it is noted in the description of the option. The only option that is handled differently, is the name of the user's program. It is put as an option on the // EXEC SSLOADER card. It is only necessary to list this option if the user/s program was not link edited during this job run. For example: a programmer linkage edits his program with the options KEEP and RELO and with the name PROG1. He comes back the next day and executes his program by // EXEC SSLOADER(PROG 1). If he had relinked his program before executing SSLOADER he would not have needed to put the name option in.

The nine different options are:

IO, SYSUNI, SYSUNIT1=SYSUNIT2, DUMP, CO=NUM, PRIO=NUM, DBUF ROLL=SYSXXX.

Their uses are:

IO This option is designed if the user wishes to use the SPOOL facility of our system. If the user specifies 10 he can read and write just as in a normal job.

SYSUNI

This option specifies which of the system units the user wishes to use in his program. The manner in which this is done is to list the units that you wish to use in your program. They can be listed in any order you wish. For example, SYS002, SYS005, SYS001, SYS000, SYSF12.

CORE=NUM

This option is used to specify the amount of core that a program requires. For a normal one phase program this does not have to be specified, SSLOADER will calculate it and print it out in hexadecimal at the end of the program. However, multiphase programs and programs that need extra core should specify this. NUM is the number of decimal bytes that the program desires. To express the amount in hexidecimal the number should be precided by an X. SSLOADER calculates the amount of core needed by the program by adding together the program size, the common size (if any) and a buffer that FORTRAN needs that is either 360 bytes long or the blocksize of any accessed data set if it is larger than 360 bytes. The user can read for the linkage edit output of this program the number of bytes that his program will take. He could use this to calculate a correct size if buffering core is used in his program.

PRIO=NUM

This option is used if the user wishes to specify the priority of his program. NUM is the priority number which is between 1 and 10. The background has a priority of seven and the default priority is seven for SSLOADER. Therefore, the time will be split between the background and the foreground program. If the NUM is higher then 7, the foreground will get the machine if it wants it. If the user's program is CPU bound the user should give a priority of less than seven to only run when the other program is not.

DUMP

This option is used if the programmer wants a dump if and when his program abends. The core dump is only of the user's partition and a save area trace is also given. To avoid core dumps being printed on the typewriter, DUMP can only be given if the user has SYSLST assigned to something or has specified 10.

PGM=PROGNAME

As an alternative to listing the name of the program to be executed as an option on the SSLOADER execute card, it can be specified as a parameter on the option card along with the other options.

OPTIONS' OPTION1 (, OPTION2, . . .)

With the executing phase that will run in the background, it is possible to list after the phase name on the// EXEC card a series of from one to six options that will be stored in the user comm region, where the user can refer to them later. Users commonly do this when they put Ma or NOSOURCE on a FORTRAN statement. To send this option to a foreground or roll job the user must use this options parameter. The user can list from one to six parameters here and they will be placed in the user comm region.

ROLL=SYSNNN (Where SYSNNN is accessed to SDSROLL)

This is used for those jobs that are going to be rolled in and out. If this is specified the following options are ignored:SPOOL, PRIO, and DUMP. Also the second sysunit is ignored in the SYSUNIT=SYSUNIT parameter sequence. If this is specified BQROLL is called instead of CATTAC.

DBUFF

This parameter is used to ask for double duffering core space to be added to the end of the partition. The sysnunits that are specified after this parameter will be given 2 times their blocksize in core. The default options of SSLOADER and the options that you can't change are the number of sysunits, temporary partition, no spool, no dump, a 14 word user comm region, a transient region, prio of seven, no save area trace and floating point register save area. SSLOADER also prints out error messages that either it, CATTAC, or the ROLLER senses. For most of the messages a self-explanatory message is given. Errors most often seen are:

ILLEGAL NOMENCLATURE - the user put some meaningless gibberish on an option card.

SEVERITY TOO HIGH - there were some high severity errors in previous job steps in this job.

PRIORITY TOO HIGH - the user asked for a priority higher than is allowed.

NOT ENOUGH CORE FOR PARTITION - the users program will not fit into the core available for a partition.

F. "ROLLIN ROLLOUT WRITE-UP"

This write-up discusses the rollin-rollout subsystem that has just been added to P. S. We will briefly describe what the roll system is, how to use it, and give a vague idea of how its works.

Who Can Use the Rollin-Rollout System

Basically the roll system is designed to process long jobs (from 10 minutes to 10 hours) in idle time between regular jobs. The roller saves its subject program on disk whenever a normal P. S. background or foreground job appears and rolls its subject job into core when there are no more P. S. jobs, or when the roll job will fit in core with the foreground job. Since the machine is idle 90% of the time anyway, we could really make a lot of idle time available to long users. There are a few requirements on jobs eligible for the roller. Obviously, the job should be intended to take a long time. Other restrictions are:

1. The job cannot use the 1827 or the Adage. This is not much of a restriction anyway since real time ADC-DAC programs couldn't use the roller.

2. Since as will be seen, roll jobs are not terminated by job control, roll jobs can not update the "last block written" pointer for a disk data set. Refer to the write-up on CATTAC for a complete discussion of this restriction. Simply put: (Input data sets don't matter) output disk data sets should be on FMT data sets only.

3. The program shouldn't use several private disk packs and tape volumes since the roll job would tie up several valuable drives all day long. This is largely a matter of courtesy and a little common sense should help here. For example, it wouldn't be nice to read five records from a reel of tape in a job that runs for 2 hours. You would probably end up blocking the drive for say 3 hours. In a simple case like this you should first copy your tape to a disk data set on a permanent pack in a normal P. S. job and then make your long roll run.

4. Your timer for SVC SETIME is not stopped while you are rolled out of core (System Programmer Note-this should be easy to change.)

5. You may not use the unit you assigned to the roll data set.

6. Multi-phase programs must use KEEP in their linkedits (you do not need relo as foreground jobs do).

How To Run a Roll Job

Roll jobs are set up almost like foreground jobs are. You put a special code on your JOB card (a 4 in column 78), do any compile and link edit steps you need and then:

```
//SYSxxx ACCESS SDSROLL
// EXEC SSLOADER
<parameter cards>
/*
<data cards>
/& END OF JOB
```

You specify on the parameter cards what I/O units you want and the name of your program. You may get a complete description of SSLOADER elsewhere. Here we just note that only the parameters OPTIONS=, PGM=, ROLL=, and SYSxxx are used by the roll jobs. PRIO=, SPOOL, DUMP are ignored. Furthermore, I/O units may not be reassigned (as they can in foreground jobs) so you only need to list those you want in the form; SYSxxx, SYSyyy, SYSzzz, . . . rather than SYSxxx=SYSaaa, etc. Do not list the unit you assigned to SDSROLL here, it is off-limits to your program. The time you have left in your job will be multiplied by 10 (to give you a realistic set of running times). Thus, if you request 10 minutes on your job card, and your compilation takes 1 minute, you will be given 90 minutes.

G. Fortran Disk I/O

Disk I/O can be performed in Fortran in two distinct manners. The first and simplest method is to treat the data set as a sequential (tape-like) file. That is, data blocks are not referenced by a block number, but are referenced sequentially. In this method, the programmer uses standard READ and WRITE statements containing an appropriate data set reference number. In addition, REWIND, END FILE, and BACKSPACE statements are used for control operations.

The second method is to use a DEFINE FILE statement and direct access READ and WRITE statements which are discussed in the IBM publication "Fortran IV Language." This method allows both sequential and random access of the data set. When using this method, difficulties may occur if the following restrictions are not observed:

1. Do not attempt to read or write more than one block of data with a single READ or WRITE statement.

2. The data set must have been created with FMT option on the ALLOC card.

Two system data sets, SDS000 and SDS001 are available for temporary storage. Since these data sets are used by the assembler and Fortran compiler, they cannot be used for permanent data storage. The data set SDS000 consists of 1000 blocks of 792 bytes (a total of 360,000 bytes or 90,000 words), and SDS001 consists of 3500 blocks of 1680 bytes (a total of 1,260,000 bytes or 315,000 words.

H. How to List a Deck (See Figure 2A)

In order to list your deck, use the following format and card order:

1. //NAME JOB, LIST (white control card).
2. Cards to be listed.
3. /& END OF JOB (blue control card).

I. How to Put a Program on a User Library

In order to produce an object deck, use the following format and card order:

1. //NAME JOB, xxxxxx(white control control card) where xxxxxx is your six-digit job number.
2. // ACCESS USRLIB where USRLIB is the name of your data set.
3. // DELETE USRLIB (prog) where prog is the name of the subroutine.
4. //SYS000 ACCESS USRLIB(prog),NEW
5. //MAIN44 EXEC FORTRAN(red control card).
6. Source Deck.
7. /& END OF JOB (blue control card).

4. PROGRAMMING AND DEBUGGING HINTS

A. System Notes for Users

The following notes and guidelines concerning the 360/44 system are of great importance.

1. Fortran data set reference number 3 refers to SYS005 instead of SYS003.
2. Default operations for "// EXEC FORTRAN"are: NODECK, SOURCE, LINK EBCDIC, NOMAP. Default operation is a process which is the reverse of an option. For example, the default option of DECK, which produces a module deck on SYSPCH, is NODECK, which does not produce a module deck.
3. Default operations for "// EXEC ASSEMBLE" are: LINK, NODECK, LIST, NOXREF.
4. Default operations for "// EXELCRLNKEDT"are: NOKEEP, NOMAP.
5. Default operations for "// EXEC LOADER" are: NOKEEP, NOMAP, LINK.

B. Fortran Options and Parameters

The following is a list of the Fortran parameters and options, some of which can be used for locating and debugging your errors.

NOTE: Parameters underlined are default options.

Parameter	Reason for Specifying
DECK	To produce a module deck on SYSPCH.
NODECK	No deck produced.
NO SOURCE	To suppress production of a source listing.
SOURCE	To produce a source listing
NOLINK	To suppress the writing of the module on SYS000, the linkage editor I/O.
LINK	Module written on SYS000.
BCD	Required if any source statements are punched in BCD (i. e. , if source cards are punched on a Model 026 key punch).
EBCDIC	Default option (for cards punched on a Model 029 key punch).
MAP	To produce a compiler storage map on SYSLST.
NOMAP	No compiler storage map produced.

In order to use any of the above parameters for debugging, the parameter should be indicated after the EXEC FORTRAN statement within parentheses (red control card). For example if you want the compiler storage map, the red control card should read:

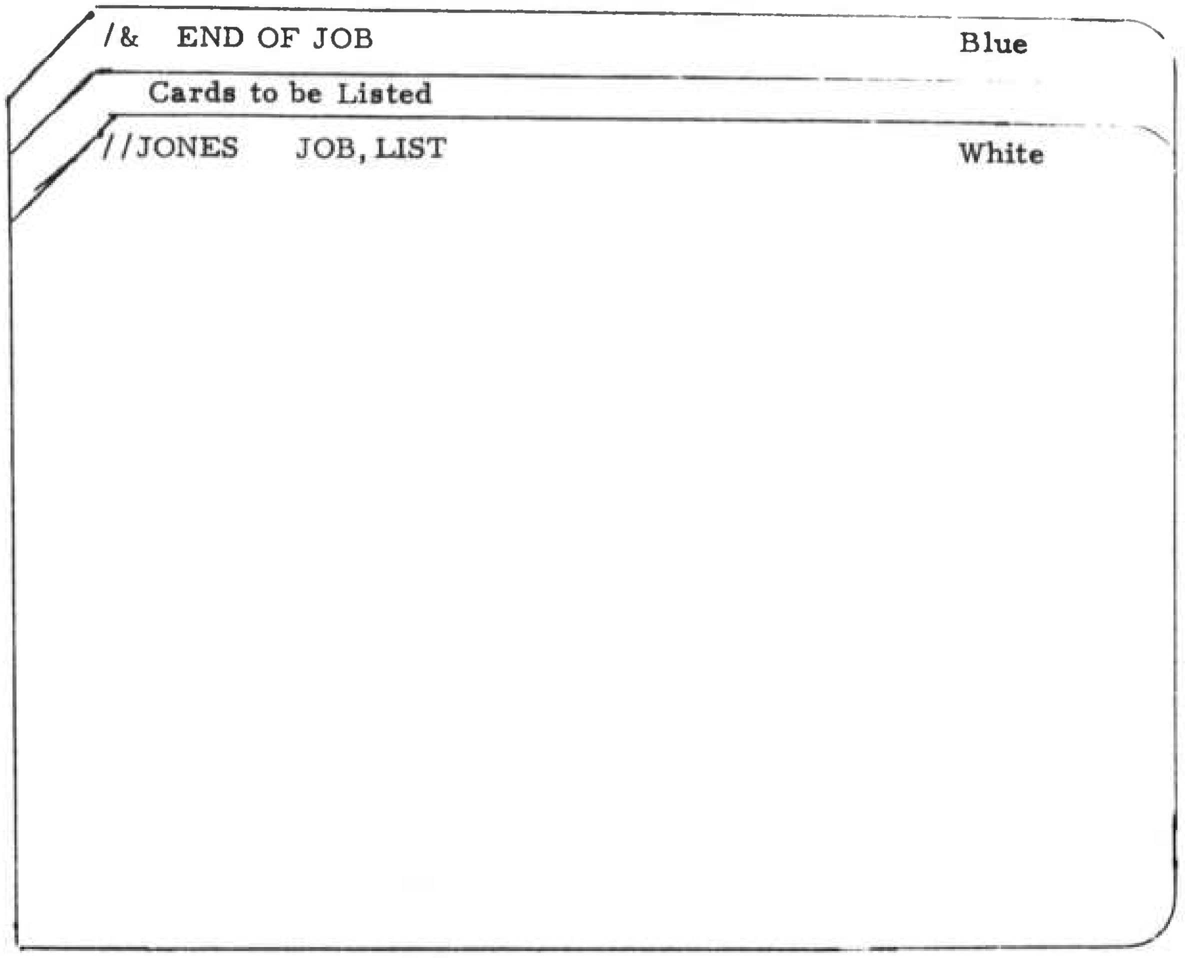
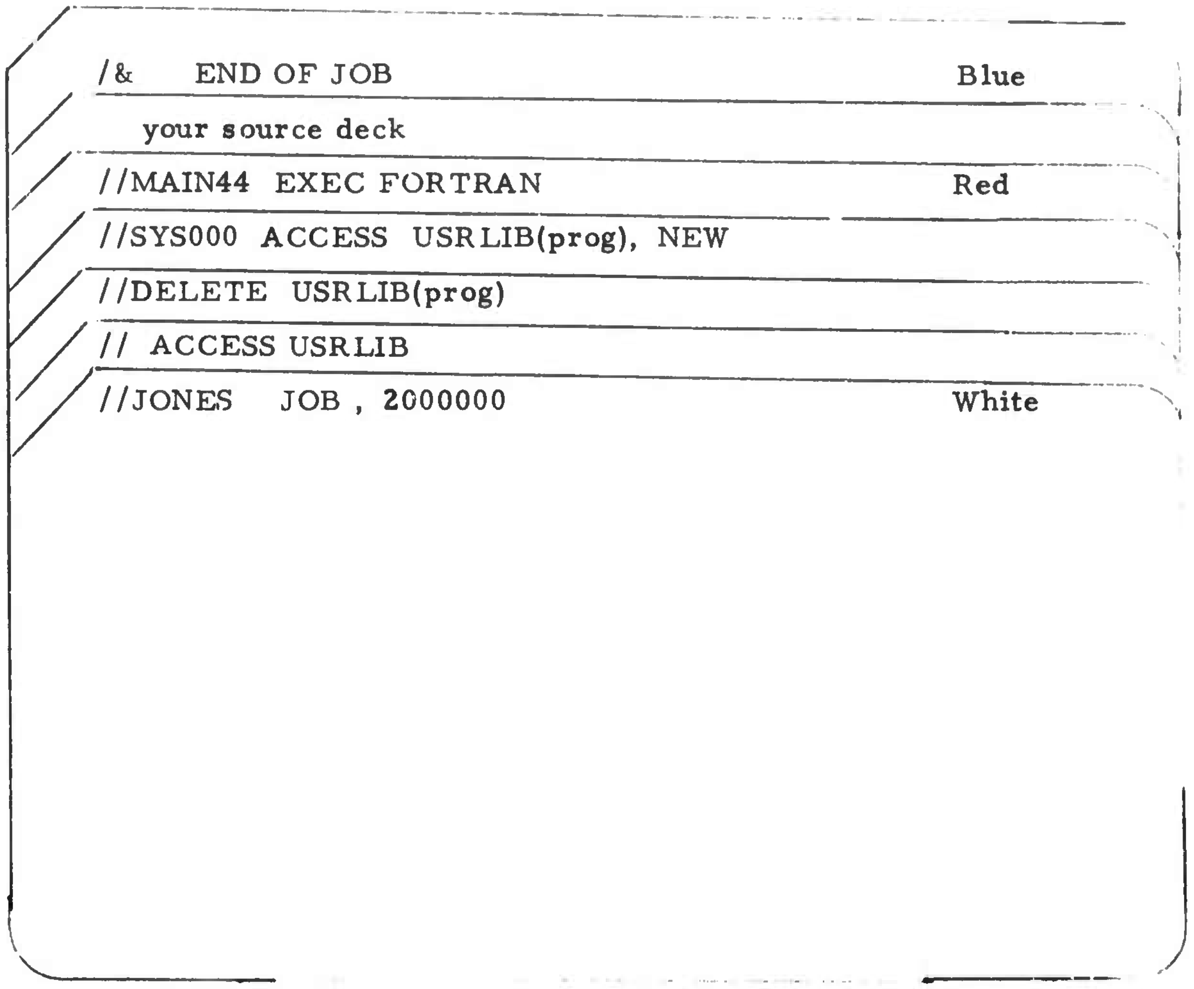


Figure 2A. Deck Set Up for Listing



```
/&      END OF JOB                                   Blue
   your source deck
//MAIN44  EXEC FORTRAN                               Red
//SYS000  ACCESS  USRLIB(prog), NEW
//DELETE  USRLIB(prog)
//  ACCESS USRLIB
//JONES    JOB , 2000000                             White
```

Figure 2C. Deck Set Up for Putting a Program on a User Library

```
//MAIN44 EXEC FORTRAN(MAP)
```

Note; The first step in find the location of your error is to get a map of the compiler and loader storage. To get the map of loader, the yellow control card should read:

```
// EXEC LOADER(MAP)
```

C. How to Read the Control Cards

Each control statement contains from 1 to 5 fields. These fields are:

1. Identifying characters field.
2. Name field.
3. Operation field.
4. Operand field
5. Comments field.

The first tow columns of any job control statement must contain the identifying characters. These are "/&" for the end of job statement, "/*" for the endo of data statement, "*" followed by a blank for the comments statement, and "//" for all other job control statements.

If a name field is used in those statements which are permitted to have one, it must start in Column 3. If the name field is not used, the third column must be blank. In all other statements, except comments, the third column must be blank.

The operation field may start in any column after Column 3, but it must be preceded and followed by at least one blank.

Similarly, the operand field may start in any column, but it must be preceded and followed by at least one blank. Parameter options, enclosed in parentheses, are part of the operand field. No blanks may preceded the left parentheses.

5. OPERATING SCHEDULE AND PRIORITIES

The computer lab operates on an open shop basis and any person with a valid job number may use the computer.

SCHEDULE

The open shop hours are as follows:

M, TU, W, TH	8:00 AM to 11:00 PM
Saturday	10:00 AM to 6:00 PM

RUNNING TIME LIMITATIONS

During open shop hours, no job is allowed to run in background over 10 minutes.

PRIORITIES

Funded users can use the lab on Sundays and can reserve larger blocks of time (an hour or more) through the lab manager. For further details contact the lab manager.

OVERNIGHT JOBS

Jobs may also be left to run overnight. When leaving jobs in the overnight box, use the following procedures:

1. Assemble your deck according to Figure 1.
2. Use the card, CALL $ TIME $(MIN), as the <u>first executable statement</u> of your program.

The $ TIME $ Routine and its usage is explicitly described in the Subroutine Section of this Manual (page 21).

6. LABORATORY REGULATIONS AND GUIDELINES

The following guidelines concern every user of the Engineering Computer Laboratory. Your commitment to these regulations would be appreciated.

1. No one except the operator on duty is allowed to operate the IBM 360/44 computer and its peripheral equipment (with the exception of the card reader).

2. Only the operators on duty and working staff programmers are to be in the console area.

3. For operating the keypunches ask for the assistance from the operator on duty (only when he is not busy).

4. When you are finished with punching cards, and before leaving the keypunch, make sure that the card punch area is clean and your discarded cards are stacked in the used-card files (located to the right of the entrance, on top of the keypunch and the other on top of the output shelf).

5. Since space is limited in the lab, after running your program use the user work room or an empty classroom for manual debugging, corrections or modifications to your program.

6. Put unwanted printouts, folded neatly, in the trash cans provided.

7. Use ashtrays for cigarette butts, or else you may be asked to leave the lab. There is NO SMOKING in the computer room.

8. If you have any complaints regarding processing of your job, bring it to the operator's attention and he will take the necessary action. If further questions remain, see the lab manager.

VII. FACILITIES

The Systems Simulation Laboratory is equipped with a general purpose hybrid computer system. The major parts of the system include an IBM 360/44 digital computer with 64K words (256K bytes) of core memory of which approximately 50K words are available to Fortran users, two 2315 disk drives, 5 2314 disk drives, a card reader and a line printer, two 3420 tape drives and an Adage 100KC A-D and D-A converter. Other ECL facilities include a FM tape recorder and an AGT-10 graphics computer.

VIII. SUBROUTINES

The subroutines listed in this section were specifically written for the lab by the staff of the engineering computer laboratory, and are kept on the Systems Library.

The list and location of the available scientific subroutine package (SSP) provided by IBM can be obtained from a staff programmer. Individual scientific subroutines or a combination of them can be used to carry out numerous functions in statistics (multiple linear regression, canonical correlation, random number generation, etc.) or mathematics (matrix arithmetic, solution of systems of first-order differential equations, Bessel function evaluation, etc.).

A. DATE

PURPOSE

This subroutine makes the date available to a user's program.

USE

This program is available as a Fortran subroutine by using:

```
CALL DATE (I, J)
```

The arguments are fixed point variables. After the CALL, they will contain the date, which can be printed with a 2A4 format specification. This subroutine requires 80 bytes, and its execution time is 379 usec.

EXAMPLE

```
        CALL DATE (I, J)
        WRITE (6,  100) I, J
    100 FORMAT (10X, "DATE IS", 2A4)
```

B. CLOCK

PURPOSE

This subroutine reads the contents of the internal real time clock.

USE

This program is available as a Fortran subroutine by using:

```
        CALL CLOCK (I)
```

The argument is a fixed point variable. After the CALL, it will contain the number of seconds since "midnight", in units of 1/60 second. This subroutine requires 74 bytes, and its execution time is 395 usec.

EXAMPLE

```
        CALL CLOCK (M)
        .
        .
        .
        CALL CLOCK(MM)
        Z = MM-M
        TIME = Z/60
```

D. TRACE, UNTRAC

The TRACE routine interprets the subject program and prints out an instruction-by-instruction report of its execution. This includes the instruction being executed, register contents before and after each instruction, contents before and after each instruction, contents of storage, setting of condition code, etc. The routine properly simulates supervisor calls (SVC) of the subject program. To call TRACE in assembly language the subject program must place the entry address of TRACE in register 15 and the address of the first instruction to be traced in register 14. It them branches to the address in register 15.

EXAMPLE

```
      . . .
      EXTRN TRACE
      L 15, = A(TRACE)
      BALR 14, 15
      . . .
```

The tracing starts at the instruction immediately following the BALR. The subject program is free to use registers 14 and 15 for any purpose after entering the trace.

The trace program running under PS correctly executes all problem state instructions. To exit from the trace and return to normal program execution use

```
      . . .
      SVC 255
      . . .
```

This will cause the trace to restore all registers to their current status except 15 which is used as a return register and hence contains the address of the instruction immediately following the SVC 255 signal.

To use the trace in FORTRAN use

```
      . . .
      CALL TRACE
      . . .
```

To return use

```
      . . .
      CALL UNTRAC
      . . .
```

NOTES ON USE OF TRACE

In the print out branches that are taken are indicated by a line skip. Some program interrupts in the subject program will cause program interrupts in the trace; a dump following such an interrupt will not show the correct GPR contents (for subject program) and the OLD PSW address will refer to a location in TRACE, not the subject program.

E. RANDOM NUMBER GENERATORS

There are two FORTRAN callable random number generators, a uniform generator and a Gaussian generator, in the library data set SDSREL.

RANDM

This is a function of one argument that generates floating point numbers uniformly distributed between 0.0 and 1.0 exclusively. It is an efficient implementation of the subroutine RANDM described in the manual <u>System/360</u>

Scientific Subroutine Package Form "H20-0205-2".

The format of the Fortran call is

```
          X = RANDM(IRAND)
```

The parameter IRAND is used as a seed to generate a random sequence. It should be initialized as an odd integer with nine or less digits. IRAND will be changed by RANDM each time it is called.

EXAMPLE

To compute uniformly distributed random numbers between -5.0 and 0.0 use

```
          DATA IRAND /11111/
                 :
     1    X = RANDM(IRAND)*5.0-5.0
```

Each time statement 1 is executed X will be set to a floating point number between -5.0 and 0.0, and IRAND will be set equal to a random odd positive integer between zero and 2**31.

Storage requirements: 96 bytes

Execution time: 120 microseconds

RANDG

This is a floating point function requiring one parameter and computes random numbers normally distributed with zero mean and a standard deviation of one. This routine is a highly efficient implementation of the subroutine RANDG described in the manual System/360 Scientific Subroutine Package Form "H20-0205-2".

Format of the FORTRAN call is

```
          X = RANDG(IRAND)
```

The parameter IRAND is used as a seed to generate the random sequence and should be initialized as an odd integer with nine or less digits.

EXAMPLE

To generate random numbers with a Gaussian distribution with a mean of 0.707 and a standard deviation of 2.0 use

```
          DATA IRAND /11111/
          . . .
     1    X   = RANDG(IRAND)*2.0+0.707
          . . .
```

Each time statement 1 is executed X will be set equal to a random number normally distributed and IRAND will be set equal to a positive random integer between 1 and 2**31.

Storage requirements: 136 bytes

Execution time: 560 microseconds

F. BIT MANIPULATION ROUTINES

The IBM-360 word length is 32 bits. The following Fortran callable functions can be used to manipulate these bits. For a detailed explanation of the operation of any of these functions see "IBM 360 Principles of Operation." The package requires 192 bytes of storage.

IOR (360 OR instruction)

An integer function of two arguments that returns the inclusive OR of its arguments.

Example:

```
I = IOR (5, 12)
'I' will equal 13.
Time: 32 microseconds
```

IAND (360 NR instruction)

An integer function of two arguments that returns the logical product of its two arguments.

Example:

```
I = IAND(12, 5)
'I' will equal 4
Time: 32 microseconds
```

IXOR (360 XR instruction)

An integer function of two arguments that returns the exclusive or of its arguments.

Example:

```
I = IXOR(12, 5)
'I' will equal 9
Time: 32 microseconds
```

ISRL (360 SRL instruction)

An integer function of two arguments. The result is the first argument right shifted the number of bits (modulo 32) specified by its second argument. The second argument must be a positive integer or zero. Zeros are shifted in for vacated higher order bits (notice that this differs from division by powers of 2).

Example:

```
I = ISRL(13, 1)
"I" will equal 6
Time: 32 microseconds
```

ISLL (360 SLL instruction)

This function is similar to ISRL except now the first argument is left shifted the number of bits indicated by the second argument (identical with multiplication by powers of 2).

Example:

```
I = ISLL (13,2)
"I" will equal 52.
Time: 32 microseconds
```

9. ERRORS AND ERROR MESSAGES

The indication of the occurrence of errors in your program is an alpha-numeric statement called the error message. Errors basically occur during two phases:

a. Compilation
b. Execution

In the case of a compilation error, a dollar sign ($) may appear under the character(s) in error with an error message following afterwards. In the case of an execution error, the error message OA210I - PROGRAM INTERRUPT ( )- OLD PSW is xxxxxxxyxxzzzzzz, is printed out.

A list of error messages is given at the end of this section.

For an interpretation of PSW errors, see error message OA21CI as well as the section entitled, DETERMINING LOCATION OF ERRORS.

ERROR MESSAGES

OA200I - An attempt was made to read from a output data set.

OA201I - An attempt was made to write into an input data set.

OA204I - An attempt was made to use a rewind, backspace, or end file statement on a data set described by a define file statement.

OA205I - A data set reference number less than 1 or greater than 11 has been used.

OA206I - An "ACCESS" statement is missing.

OA208I - Invalid printer carriage control character has been used or an input/output request has been made that is invalid for a data set such as read from a printer or write on card reader.

OA210I - PROGRAM INTERRUPT ( ) -OLD PSW IS xxxxxxxYxxzzzzzz- A program check occurred. zzzzzz is the location of the error (see "DETERMINING LOCATION OF ERRORS") "Y" gives the reason for the interrupt:

"F" - floating point division by zero.
"D" - floating point underflow (result <5.E+75).
"C" - floating point overflow (result >7.E+75).
"9" - integer division by zero.

"6" is caused by COMMON or EQUIVALENCE errors at compile time or by passing REAL*4 (or INTEGER*2 or COMPLEX*8) values to a subprogram which has REAL*8 (or INTEGER*4 or COMPLEX*16) arguments. This message is followed by "OA219I".

OA211I - An invalid character has been detected in a FORMAT statement.

OA212I - An attempt has been made to read or write a record that exceeds the buffer length (360 for disk or tape, 133 for printer, 80 for cards).

OA215I - An invalid character exists for input under I, E, F, or D format code.

OA217I - An end-of-data condition was sensed during a READ operation or an end-of-extent condition was detected during a WRITE operation, or a / in column one was encountered in the input data.

OA2251 An invalid character is encountered for the z format code.

OA2321 The position of a record is outside the range of the DEFINE FILE statement.

OA2411-OA2471 An attempt was made to raise zero to a negative (or zero) power

OA2511(OA2611) Square root of a negative number.

OA2521(OA2621) The argument of EXP or DEXP is greater than 174. 673.

OA2531(OA2631) The value of the argument ALOG or ALOG 10 (or DLOG or DLOG10) is less than or equal to zero, or an attempt has been made to raise a negative base to a real power.

OA2541(OA2641) The absolute value of an argument of SIN or COS DSIN or DCOS is greater than or equal to . 8235E+06 (or. 3537D+16 for OA2641).

OA2941 More than 50 program checks have occured.

OA2991 More than 50 pages of Fortran print has been produced.

PRINTED RESULTS ARE SMALLER THAN 1. 0E-79-An INTEGER number was written under E format.

PRINTED RESULTS ARE MULTIPLIED BY A POWER OF 10 - This is sometimes caused by incorrect use of P format. When using P format the P specification applies until the end of the read or write statement or until the next P.

****IN PRINTED OUTPUT - An attempt was made to print a number greater than a format specification could handle. This could occur if your program calculates a number too large or you try to print a REAL number under "I" format.

SOURCE OF ERRORS - The first two letters of each message identify the portion of the system which detected the error. They are as follows:

FA, FB, GA, HA, JA - supervisor
IA - Job Control processor
KA - linkage editor
NA - Fortran Compiler
OA - execution time Fortran routines

FAOFI 00C0 I/O PROG CHK - Program too big for memory

FAOFI 00E I/O PROG CHK - Invalid printer control character

FBOBI OPRTR CHCLED - A job has been canceled by the operator

GA081xxxxxxxxCAN'T BE FTCHD - xxxxxxxx was used as the name of a program, and the system cannot find any program with this name in the phase library, or the program was too big for memory.

HA031 DIMENSION EXCEEDED - This message is preceded by "PROG CHK INT CODE Y". If Y is 1 no other information is available from this message. If Y is 5 use the last six characters from the OLD PROG CHECK PSW as "zzzzzz" to determine the statement that caused the error. (See "DETERMINING LOCATION OF ERRORS").

IAO11STMNT FMT ERR - This message usually appears when a program did not read all its data and a data card was mistaken for a job control statement, often because of an error in a format statement or the job ended early because of an "OA" message The line printed is the card after the last one read by your program.

IAO11STMNT FMT ERR - This message usually appears when a program did not read all its data and a data card was mistaken for a job control statement, often because of an error in a format statement or the job ended early because of an "OA" message. The line printed is the card after the last one read by your program.

IA02I STMNT FMT ERR - Something other than a symbolic unit name is specified in the name field of an ACCESS statement. See also "IA01I".

IA03I STMNT FMT ERR - An invalid operation was specified. See also "IA01I".

IA04I STMNT FMT ERR - A required parameter is missing. See also "IA01".

IA05I STMNT FMT ERR - An improper character has been used. See also "IA01I".

IA14I STMNT SEQ ERR - An error was detected during compile or link edit so that the program cannot be executed.

IA17I STMNT SEQ ERR - Invalid job control statement. See also "IA01I".

IA23I VOL REQ ERR - The volume field of an ACCESS statement is incorrect.

IA32I DSNAME ERR xxxxxxxx - The required member is not in the data set specified.

IA33I DSNAME ERR xxxxxxxx - The data set named cannot be found in the system catalog.

IA35I DSNAME ERR xxxxxxxx - xxxxxxxx is a duplicate member name.

IA43I INSUFF SP xxxxxxxx - xxxxxxxx is the name of a directoried data set whose directory is full.

IA44I INSUFF SP xxxxxxxx - xxxxxxxx is the name of a directoried data set in which there is not enough room to add another member, or it is the name of a data set of any type in which there is not enough room to write another block of data.

IA50I ABN EOJ - The job did not include a /& (end-of-job) statement.

IA58I CUU RW RR RN PW PR PN - An input/output error occurred. A count exists under each type listed in this message. Recovered errors caused no problems. Permanent errors may produce undesired effects. CUU is the device address; RW is the number of recovered write errors; RR, recovered read errors; RN, the number of recovered nondata transmit errors; PW, permanent write errors; PR, permanent read errors; and PW the number of permanent nondata transmit errors.

IA89I M cuu volid - The system has requested the operator to mount the disk which contains the data set named in an ACCESS statement.

IA91D - VOL xxxxxx MIS-MOUNTED - The disk pack requested by your ACCESS statement was not mounted at the beginning of your job. The operator will mount your disk and your job will continue.

JA0AI JOB CANCELED - A job has been canceled. Another message usually appears giving the reason for the cancelation.

KA06I - A job control statement other than /* was read. It has been saved for processing at the end of the job step.

KA38I - A "/*" card does not follow the "// EXEC LNKEDT" card. If the PROCLIB (i.e. FORTX etc.) is used there should be two /* cards between the Fortran program and the data. If linkage editor control cards are used, there is a misspelled key word or module decks do not precede phase and include cards.

KA42I - An attempt was made to initialize a variable in common.

KA61I - Program too big for memory.

KA71I xxxx UNRESOLVED ADDRESS CONSTANTS - The xxxx field is replaced with the number of unresolved external references (functions and subroutine). A list of unresolved symbols is written. These subprograms are not available in the relocatable library, or were not supplied in this job. A subprogram name may be misspelled, or a subscripted variable may not be dimensioned, or an error in compilation occurred causing the subroutines that follow to be ignored.

KA80I 2 - The program is too big for memory or there is not enough room to keep the program on disk.

KA88I - The EXEC statement name field was balnk when the module was complied or errors occurred during compilation.

KA91I - A program with the same name is already in the phase library.

NA01I ILLEGAL TYPE - The variable in an Assigned GO TO statement is not an integer variable; or, in an assignment statement, the variable on the left side of the equal sign is of logical type and the expression on the right side is not; or, the arguments of a Fortran library function are the wrong mode.

NA02I LABEL - A statement that should have a statement number does not. For example, a FORMAT statement or a statement following a GO TO statement is not labeled.

NA03I NAME LENGTH - The name of a variable, COMMON block, NAMELIST, or subprogram exceeds six characters.

NA04I COMMA - A comma required in a statement does not appear.

NA05I ILLEGAL LABEL - Invalid use of a statement number has occurred; for example, an attempt has been made to branch to a FORMAT statement.

NA06I DUPLICATE LABEL - This statement number has previously been used.

NA07I ID CONFLICT - The name of a variable or subprogram has been used in conflict with the type that was defined for it in a previous statement. The name listed in a CALL statement is the name of a variable, not subprogram, or a single name appears more than once in the dummy list of a statement function.

NA08I ALLOCATION - The storage assignment specified cannot be performed because the use of a variable name is in conflict with some prior use of that name. A name listed in a COMMON block has been listed in another

COMMON block; a variable listed in an EQUIVALENCE statement is followed by more than seven subscripts; in a SUBPROGRAM a variable is passed as an argument and is in common.

NA09I ORDER - Source statements are used in an improper sequence. For example, an IMPLICIT statement appears as other than the first statement.

NA10I SIZE - The size specification is an explicit type statement is not one of the acceptable values or an integer constant is too large.

NA11I UNDIMFNSIONED - A subscript has been used but the variable has not been dimensioned.

NA12I SUBSCRIPT - The number of subscripts used does not agree with its dimensions.

NA13I SYNTAK - A statement cannot be identified, a nondigit appears in the label field, fewer than three labels follow the expression in an Arithmetic IF statement, a constant that begins with a decimal point does not have a digit as its second character, invalid character in a FORMAT statement, wrong count in H-type FORMAT specification, missing apostrophy in a literal specification, or any other punctuation error.

NA16I ILLEGAL STA - The context in which a statement has been used is invalid. For example, the statement in a Logical IF (the result of the true condition) is a Specification statement, a DO statement, or another logical IF statement; or an entry statement appears in a main program.

NA18I NUMBER ARG - A library subprogram is used with the wrong number of arguments.

NA19I FUNCTION ENTRIES UNDEFINED - The program being compiled is a FUNCTION subprogram, but there is no scalar with the same name as the FUNCTION.

NA21I UNCLOSED DO LOOPS - One or more DO loops are initiated, but their terminal statements do not exist or are in the wrong place.

NA22I UNDEFINED LABELS - Statement numbers used are not defined; for example there is a "GO TO 5" statement and statement 5 is missing.

NA25I DUMMY DIMENSION ERRORS - Variables specified with dummy array dimensions are not arguments of the subprogram.

DETERMINING LOCATION OF ERRORS (All arithmetic base 16)

1. Locate the module in which the error occurred: Get a loader map (or link edit map) then look in the column marked "loaded" for the largest number less than zzzzzz. The name opposite this number is the program name. Subtract the two numbers.

2. A Fortran map will help locate the statement in error. Examine the label map of the appropriate module. Find the locations in the label map nearest the result of 1. The error occurred between the associated labels. If a Fortran

map is not available it is necessary to guess about where the check occurred. A little common sense willooften save a run, e.g. a divide check can only occur in a statement with a divide. Furthermore, if there is a program check interrupt under the following format:

```
PROGCHK PSW - xxxxxxxYxxzzzzzz
```

where zzzzzz is the location of the error as before, then "Y" gives the reason for the interrupt

"1" - operation
"5" - addressing
"6" - specification

The cause of program check is usually due to subscripting, mode or boundary alignment.

EXAMPLE FOR PROGRAM CHECK ERRCR

```
      DIMENSICN A(100)

 9    Y = A(10)
10    X = A(1000000)
11    Z = A(20)
      STOP
      END
```

The error message caused would be:

```
PROG CHK PSW - FF050005 9200753A
```

where in this case an addressing error has been made, hence Y = 5. The address of the location this request was made is 00753A. The loader map gives us the following information:

```
MAIN44 Loaded 0072A8
High Core     007583
```

So 00753A is somewhere between MAIN44 and High Core. To find the absolute address we carry out the subtraction:

```
753A
72A8 -
0292
```

By having the absolute address 0292, we look under the Label Map and find:

Label	Location	Label	Location	Label	Location
a	00027E	10	000286	11	000296

Hence 000292 is somewhere between label 10 and 11. Since there are no statement numbers between 10 and 11, the error must be associated with statement number 10.

10. GLOSSARY AND ABBREVIATIONS

ACCESS: Command or statement to permit access data set.

Assemble: Translate from symbolic language to machine language, from source to object form.

Block (records):

1. To group records for the purpose of conserving storage space or increasing the efficiency of access or processing.
2. A physical record so constituted.

Condense: For directoried data sets, to shift the members and, separately, their directory entries, maintaining their original sequence, to fill space vacated through the deletion of other members. After condensing, members occupy contiguous locations. Similar to the squeeze function for direct access volumes.

DELETE: Command or statement to remove data set or member(s).

Dump (main storage):

1. To copy the contents of all or part of main storage onto an output device, so that it can be examined.
2. The data resulting from action described in 1.
3. A routine that will accomplish the action described in 1.

EXEC: Command or statement to define start of job step.

FORTRAN: FORmula TRANslation.

Hex: Hexadecimal (base 16) number system.

I/O: Input/Output unit or device.

Job An externally specified unit of work for the computing system from the standpoint of installation accounting and system control. A job consists of one or more job steps.

Job card: First card of job deck.

Job Control: Part of control program that prepares job for running. Functions include: assigning I/O devices, printing job control statements, and fetching first program phase of each job step.

Job Control Statement: Any one of the control statements in the input stream that identifies a job or defines its requirements.

Language Translator: General term for assembler or compiler that accepts statements in one (symbolic) language (source program) and produces an equivalent machine-language instruction (object) program. The 44PS has two language translators: assembler and FORTRAN.

Library: A collection of objects associated with a particular use and having a directory to locate individual objects. In this context, see module library, phase library.

Linkage: The means by which communication is effected between two routines or control sections.

Linkage editor: A program that produces one or more program phases by transforming relocatable modules into a format that is acceptable to the program fetch routine, combining separately produced modules, replacing, deleting, and adding control sections as requested, and resolving symbolic cross-reference among them.

Load:

1. Generally, to read a phase into main storage.
2. Program load -- to read a phase into main storage, and return control to the invoking program.
3. The routine that accomplishes the above two steps.

Main Storage: All addressable storage rom which instructions can be executed or from which data can be loaded directly into registers.

Module: The unit of output from a single execution of the assembler or compiler, in relocatable form and consisting of one or more control sections with control information to permit relocation and symbolic cross-references to other modules.

Module Library: A directoried data set containing selected modules and serving as an automatic source of input to the linkage editor.

Name: A set of one or more characters that identifies a statement, data set, module, phase, etc., and that is usually associated with the location of that which it identifies.

Object: Language, program, or deck, characterized by machine, rather than symbolic language, as after assembly.

Phase: The unit of output of the linkage editor, in absolute form, that is loaded by a single program fetch or program load operation; may represent an entire program or part of a program.

Phase library: The directoried data set that contains program phases, processed and entered by the linkage editor; the source from which program phases are loaded for execution.

Record: Group field(s) of data as logical entity (e.g., logical data record = punched card). On tape, records are blocked, thus defined as record-block.

Relocatable Form: A form of program text wherein the instructions have variable load addresses and symbolic cross-references, plus control information to permit later conversion to absolute form.

Relocation: The modification of address constants required to compensate for a change of origin of a module or control section.

Subroutine: A routine that perfroms a specific subordinate function that is part of the overall objective of a larger routine.

Supervisor: As applied to the Model 44 system, the routines executed in response to a requirement for altering or interrupting the flow of operations through the central processing unit, or for performance of input/output operations, and, therefore, the medium through which the use of system resources is coordinated and the flow of operations through the central processing unit is maintained.

Symbol: In programming, code (generally mnemonic) in non-machine language In addressing, code for undetermined absolute address In I/O unit identification, code for function, permitting the system to address suitable device to perform job. Symbolic units are identified by the first three letters: SYS.

User: Anyone who requires the services of a computing system.

Utility Programs: A collection of programs (together, the utility processor) that perform volume initialization and maintenance and data set transmission functions.

Word: Predetermined group of bytes, using the first byte-address as the word-address grouped for functional application, located in storage by boundary limits: halfword = two bytes, location divisible by 2; fullword = four bytes, location divisible by 4; doubleword = eight bytes, location divisible by 8

OA2251 An invalid character is encountered for the z format code.

OA2321 The position of a record is outside the range of the DEFINE FILE statement.

OA2411-OA2471 An attempt was made to raise zero to a negative (or zero) power

OA2511(OA2611) Square root of a negative number.

OA2521(OA2621) The argument of EXP or DEXP is greater than 174. 673.

OA2531(OA2631) The value of the argument ALOG or ALOG 10 (or DLOG or DLOG10) is less than or equal to zero, or an attempt has been made to raise a negative base to a real power.

OA2541(OA2641) The absolute value of an argument of SIN or COS DSIN or DCOS is greater than or equal to . 8235E+06 (or. 3537D+16 for OA2641).

OA2941 More than 50 program checks have occured.

OA2991 More than 50 pages of Fortran print has been produced.

OA294I More than 50 program checks have occurred.

OA299I More than 50 pages of Fortran print has been produced.

PRINTED RESULTS ARE SMALLER THAN 1. 0E-79-An INTEGER number was written under E format.

PRINTED RESULTS ARE MULTIPLIED BY A POWER OF 10 - This is sometimes caused by incorrect use of P format. When using P format the P specification applies until the end of the read or write statement or until the next P.

****IN PRINTED OUTPUT - An attempt was made to print a number greater than a format specification could handle. This could occur if your program calculates a number too large or you try to print a REAL number under "I" format.

SOURCE OF ERRORS - The first two letters of each message identify the portion of the system which detected the error. They are as follows:

FA, FB, GA, HA, JA - supervisor
IA - Job Control processor
KA - linkage editor
NA - Fortran Compiler
OA - execution time Fortran routines

FAOFI 00C0 I/O PROG CHK - Program too big for memory

FAOFI 00E I/O PROG CHK - Invalid printer control character

FBOBI OPRTR CHCLED - A job has been canceled by the operator

GA081xxxxxxxxCAN'T BE FTCHD - xxxxxxxx was used as the name of a program, and the system cannot find any program with this name in the phase library, or the program was too big for memory.

HA031 DIMENSION EXCEEDED - This message is preceded by "PROG CHK INT CODE Y". If Y is 1 no other information is available from this message. If Y is 5 use the last six characters from the OLD PROG CHECK PSW as "zzzzzz" to determine the statement that caused the error. (See "DETERMINING LOCATION OF ERRORS").

11. THE KEY PUNCH MACHINE

The ON/OFF switch for the key punch (KP) is on the right hand side of the machine, under the keyboard. Before operating the machine, set all switches. Normally all switches except the one marked CLEAR should be in the "up" position. (If there is a switch marked INTERPRET/PUNCH, it should be set to PUNCH.) Make sure that there are cards in the read hopper, and that the rocker switch under the control drum is over to the right. This disengages the control drum. If punching is to be carried out under control of the drum, the rocker switch must be in the left position.

To initialize punching, briefly lift the spring loaded switch marked CLEAR until the machine goes through at least one cycle, and then press the FEED key twice (2). This operation will bring a card into position to be punched.

The normal mode of operation of the KP is lower case symbols. To punch upper case symbols (integers, etc.), hold down the key marked "NUM SHIFT" while simultaneously depressing the desired symbol to be punched (upper case). After punching the last symbol on a card, the card can be removed from the punch area by depressing the REL (release) key.

Cards can be shifted backwards through the punch area by pressing the BACKSPACE button below the drum.

If you make a mistake punching a card, it is NOT necessary to repunch everything that preceded the mistake. Simply press REL, then check which column the error is in, and then duplicate the card up to this column and punch the correct entry. Duplication may be done by pressing the DUP key, which will duplicate the previous card until the key is released.

The column to be duplicated or punched next can be determined by inspecting the position pointed at by the red arrow beneath the drum card.

More than one character may be punched in a column by holding down the "MULT PCH" key while punching all desired characters. This technique is useful when preparing drum control cards.

It is usually easier to punch cards rapidly by using the automatic CONTROL feature of the key punch. The drum card allows for automatic duplicating of sections of a card, automatic skips to new sections, and automatic numeric shift. The operator will show you how to place a card on the drum. When the rocker arm under the drum is to the left, the operation of the KP is under the control of this card. To use this feature, one must prepare a special card (drum card) that explicitly defines the format of the cards to be punched. The relevant codes for the drum card are the following:

1) a minus sign (-) (11 punch) will start an automatic skip which will continue until a drum card column is found that does <u>not</u> contain a 12 punch (&).

2) a zero (0) will start an automatic duplicate which will continue until a drum card column is found that does <u>not</u> contain a 12 punch (&).

3) the keypunch machine under drum card control assumes numeric shift unless a 1 row punch (1) is present in a column which forces ALPHA numeric shift.

EXAMPLE

Suppose you wanted to punch simple FORTRAN statements. You may want to put a C in col. 1 to define a comment card, but cols. 2-5 should contain only numbers, col. 6 should be empty (normally), and the statement begines in col. 7. The following drum card will be sufficient for this format.

col 1	-	A	(12 and 1 punch)
col 2-5	-	12	punch (&)
col 6	-	11	punch (-)
col 7	-	1	punch
col 8-80	-	A	(12 and 1 punch)

Using this drum card, it is not necessary to press numeric shift for putting statement numbers on cards. If a card to be punched is not to have a statement number, you can immediately skip to column 7 by depressing the SKIP key.

EXAMPLE

Suppose you wish to punch cards having the following format.

Column	
1	blank or C
2-5	blank or statement number
6	blank (skip)
7-72	statement
73-75	duplicate (initials)
76-80	sequence numbers

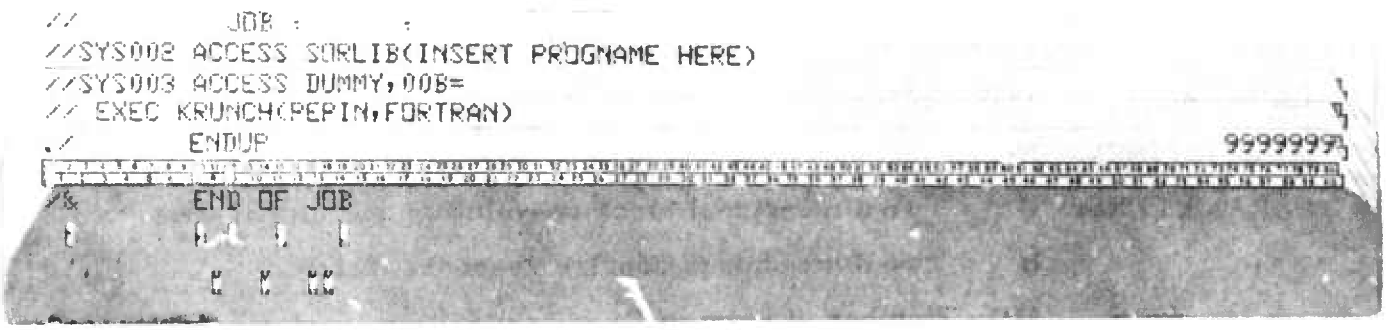
The following drum card will suffice:

Column	
1	1 punch (alpha. shift)
2-5	12 punch (numeric shift and skip if SKIP key is depressed)
6	11 punch (automatic skip)
7	1 punch (alpha. shift, end of skip field)
8-72	12 & 1(A) punch (alpha. shift and field)
73	0 & 1 punch (automatic duplicate and alpha shift)
74-75	12 & 1 punch (alpha. shift and duplicate)
76	blank (end of duplicate field)
77-80	12 punch (numeric shift field)

II - ECL SUBROUTINE LIBRARY

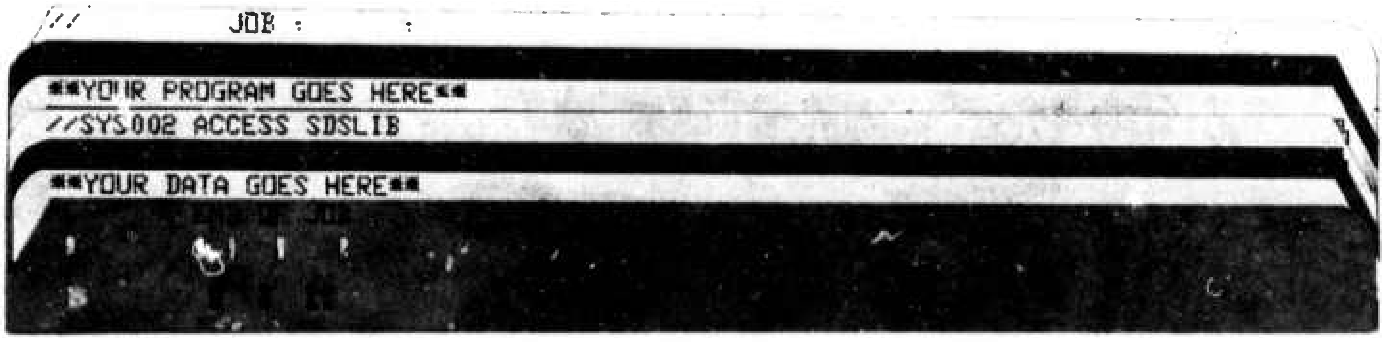
SOURCE LIBRARY (SORLIB)

The following subroutines are on SORLIB. Listings of these subroutines may be obtained as follows:



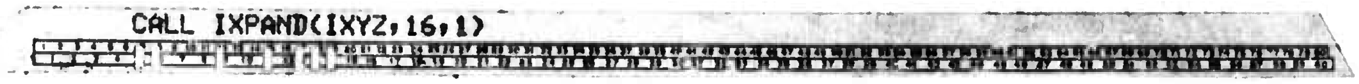
```
//       JOB :      :
//SYS002 ACCESS SORLIB(INSERT PROGNAME HERE)
//SYS003 ACCESS DUMMY,OOB=
// EXEC KRUNCH(PEPIN,FORTRAN)
./       ENDUP                                                   9999999
/&       END OF JOB
```

To use these subroutines in FORTRAN programs the following deck setup must be used:



```
//       JOB :      :
**YOUR PROGRAM GOES HERE**
//SYS002 ACCESS SDSLIB
**YOUR DATA GOES HERE**
```

In your FORTRAN program you may call these subroutines like any other FORTRAN subroutine:



```
      CALL IXPAND(IXYZ,16,1)
```

BLOKON(D, H, MB, ND, L)

TITLE:	Two Dimensional Block Convolution Operator
TYPE:	Subroutine
LANGUAGE:	FORTRAN
SUBROUTINES:	None
PARAMETERS:	D - Two dimensional block convolution operator matrix.
	H - Two dimensional impulse : esponse matrix.
	MD - Number of rows in D.
	ND - Number of columns in D.
	L - Size of impulse response.
USAGE:	This subroutine generates a two dimensional block convolution operator.

BLRKON(B, H, MB, NB, L)

TITLE:	Two Dimensional Blur Convolution Operator
TYPE:	Subroutine
LANGUAGE:	FORTRAN
SUBROUTINES:	None
PARAMETERS:	B -Two dimensional blur convolution operator matrix.
	H -Two dimensional impulse response matrix.
	MB-Number of rows in B.
	NB -Number of columns in B.
	L -Size of impulse response.
USAGE:	This subroutine generates a two dimensional circular convolution operator.

BLUR (DB, BF, ISL, ISB)

TITLE:	ONE-DIMENSIONAL BLUR CONVOLUTION MATRIX IN SINGLE PRECISION
LANGUAGE:	FORTRAN
PARAMETERS:	SAME AS THOSE OF SUBROUTINE DBLUR.
SUBROUTINES:	NONE
USAGE:	SAME AS SUBROUTINE DBLUR.

CERR (UL1, VL1, V1, UL2, VL2, V2, EL, EC)

DATE:	October 31, 1972
TITLE:	LUMINANCE & CHROMINANCE ERRORS
LANGUAGE:	FORTRAN

PARAMETERS:

UL1	-	CIE uniform chromaticity scale, u component of color #1
VL1	-	CIE uniform chromaticity scale, v component of color #1
V1	-	CIE uniform chromaticity scale, Luminance of color #1
UL2	-	CIE uniform chromaticity scale, u component of color #2
VL2	-	CIE uniform chromaticity scale, v component of color #2
V2	-	CIE uniform chromaticity scale, Luminance of color #2
EL	-	Computed Luminance Error
EC	-	Computed Chrominance Error

$$E_L = \sqrt{\frac{(V1 - V2)^2}{(V1)^2}} \qquad E_C = \sqrt{\frac{(UL1 - UL2)^2}{(UL1)^2} + \frac{(VL1 - VL2)^2}{(VL1)^2}}$$

SUBROUTINES:	NONE
USAGE:	Computes luminance and chrominance errors between two colors.

CIRKON(C, H, MC, NC, L)

TITLE:	Two Dimensional Circular Convolution Operator
TYPE:	Subroutine
LANGUAGE:	FORTRAN
SUBROUTINES:	None
PARAMETERS:	C - Two dimensional circular convolution operator matrix.
	H - Two dimensional impulse response matrix.
	MC - Number of rows in C.
	NC - Number of columns in C.
	L - Size of impulse response.
USAGE:	This subroutine generates a two dimensional circular convolution operator.

CLCON (II, IO, XI1, XI2, XI3, X01, X02, X03)

DATE: October 31, 1972

TITLE: COLOR COORDINATE CONVERSION

LANGUAGE: FORTRAN

PARAMETERS:

II - INPUT CODE (See Table)

IO - OUTPUT CODE (See Table)

XI1, XI2, XI3, - INPUT COORDINATES

X01, X02, X03, - OUTPUT COORDINATES

SUBROUTINES: MOD, MAT13

USAGE:

	I/O	COORDINATES	
CODE	1	2	3
1	R_N	G_N	B_N
2	r_n	g_n	Y
3	R_C	G_C	B_C
4	r_c	g_c	Y
5	X	Y	Z
6	x	y	Y
7	U_R	V_R	W_R
8	u_r	v_r	Y
9	Y	I	Q
10	U*	V*	W*
11	U	V	W
12	u	v	Y
13	K_1	K_2	K_3

Translate from one color coordinate system to another

CLROUT (IDSRI, IDSRO, IDSPO, ISIZE, CLIP, DOMAIN, NPIK, JOPT, NBIT)

DATE: October 31, 1972

TITLE: COLOR OUTPUT

LANGUAGE: FORTRAN

PARAMETERS:

IDSRI	-	INPUT REAL DATA SET (REAL*4)
IDSRO	-	OUTPUT REAL DATA SET (REAL *4)
IDSPO	-	OUTPUT PACKED DATA SET
ISIZE	-	NUMBER OF PIXELS PER LINE
CLIP	-	INPUT VECTOR OF MAXIMUM AND MINIMUM CLIP LEVELS FOR RED, GREEN, AND BLUE PIXELS
DOMAIN	-	INPUT VECTOR OF MAXIMUM AND MINIMUM OUTPUT LEVELS FOR RED, GREEN, AND BLUE PIXELS
NPIK	-	VECTOR OF 4 INTEGER NUMBERS (EXPERIMENT NUMBER)
NBIT	-	NUMBER OF BITS/PIXEL (1 to 8)
JOBT	-	JOB OPTION

1 - HISTOGRAM
2 - HISTOGRAM, SCALING
3 - HISTOGRAM, SCALING, PACKING
4 - HISTOGRAM, SCALING, FRAMING, PACKING
5 - SCALING
6 - SCALING, PACKING
7 - SCALING, FRAMING, PACKING
8 - FRAMING, PACKING
9 - PACKING

SUBROUTINES: DSCIO, PRESS

USAGE:

*The input and output data sets are one higher than the channel number.

//SYSØØØ ACCESS DASET

DASET is now accessed to data set #1 or IDSRI = SYSunit number + 1 and IDSRO = SYSunit number + 1.

CONTR (INPUT, IOUTPUT, ISIZE, POSLIM, ANEGLIM)

DATE:	October 31, 1972
TITLE:	MONOCHROME IMAGE SCALING
LANGUAGE:	FORTRAN

PARAMETERS:

INPUT	- INPUT UNIT NUMBER (SYS UNIT # +1)
IOUTPUT	- OUTPUT UNIT NUMBER (SYS UNIT # +1)
ISIZE	- NUMBER OF LINES IN PICTURE
POSLIM	- POSITIVE LIMIT
ANEGLIM	- NEGATIVE LIMIT

SUBROUTINES:	DSCIO
USAGE:	CLIPS & RESCALES a picture to improve grey level distribution

COSIGN:	(CIN, COUT, MC, NC, IT, IC)
TITLE:	GENERALIZED 2-DIM. COSINE TRANSFORM
LANGUAGE:	FORTRAN

PARAMETERS:

CIN	-	INPUT MATRIX
COUT	-	OUTPUT MATRIX
MC	-	NUMBER OF ROWS OF C
NC	-	NUMBER OF COLUMNS OF C
IR=1,	-	FORWARD ROW TRANSFORM
IR=-1,	-	INVERSE ROW TRANSFORM
IC=1,	-	FORWARD COLUMN TRANSFORM
IC=-1	-	INVERSE COLUMN TRANSFORM

SUBROUTINES: COST

USAGE: THIS SUBROUTINE COMPUTES THE GENERALIZED 2-DIMENSIONAL DISCRETE COSINE TRANSFORM OF A REAL MATRIX.

COST (X, N. L)

TITLE:	COSINE TRANSFORM
LANGUAGE:	FORTRAN

PARAMETERS:

X	-	input vector; also transformed output vector.
N	-	Length of vector X; must equal a power of 2.
L	-	Specifies either forward or inverse transform
		L = +1 forward transform
		L = -1 inverse transform

Usage: This subroutine computes the fast cosine transform of any vector of length equal to a power of 2.

DBLUR (DB, BF, ISL, ISB)

TITLE:	ONE-DIMENSIONAL BLUR CONVOLUTION MATRIX IN DOUBLE PRECISION
LANGUAGE:	FORTRAN

PARAMETERS:

DB	- BLUR CONVOLUTION MATRIX OF DIMENSION (ISL) BY (ISB)
BF	- GAUSSIAN BLUR VARIANCE
ISL	- NUMBER OF ROWS
ISB	- NUMBER OF COLUMNS

SUBROUTINES:	NONE
USAGE:	THIS SUBROUTINE GENERATES A BLUR CONVOLUTION MATRIX WITH GAUSSIAN-SHAPED IMPULSE RESPONSE.

DEIGEN (A, R, N, MV)

TITLE:	EIGENVALUES AND EIGENVECTORS OF A REAL SYMMETRIC MATRIX.
LANGUAGE:	FORTRAN
PARAMETERS:	SAME AS EIGEN IN SCIENTIFIC SUBROUTINE PACKAGE
USAGE:	SAME AS EIGEN EXCEPT DEIGEN IS DOUBLE PRECISION

DIST (UU, UL, XN, NI, N, CJ, EPS, ISIG)

DATE:	October 31, 1972
TITLE:	COLOR DISTANCE
LANGUAGE:	FORTRAN
PARAMETERS:	

UU	- UPPER VECTOR
UL	- LOWER VECTOR
XN	- GEODESIC
NI	- NUMBER OF INTERATIONS
N	- NUMBER OF INTERVALS BETWEEN THE TWO COLORS (THEREFORE N+1 = NUMBER)
CJ	- COLOR DISTANCE
EPS	- ALLOWABLE ERROR (SHOULD BE A POSITIVE NUMBER)
ISIG	- ISIG IS ZERO IF NO ERROR, ISIG IS ONE IF ACCURACY OF EPS IS NOT ACHIEVED WITH NI ITERATIONS.

SUBROUTINES:	COORD, SONAL, INTG, INV, TEST, DAT
USAGE:	THIS PROGRAM DETERMINES THE DISTANCE BETWEEN THE TWO COLORS XUV AND XLV(EXPRESSED IN THE UVW SYSTEM)

DMAS (A, B, C, NR, NC, IAS)

DATE:	October 31, 1972
TITLE:	MATRIX ADDITION & SUBTRACTION
LANGUAGE:	FORTRAN

PARAMETERS:

A	- INPUT MATRIX
B	- INPUT MATRIX
C	- OUTPUT MATRIX
NR	- NUMBER OF ROWS IN MATRIX
NC	- NUMBER OF COLUMNS IN MATRIX
IAS	- OPERATION (0= ADD, 1 = SUBTRACT)

SUBROUTINES:	NONE
USAGE:	DOUBLE PRECISION MATRIX ADD AND SUBTRACT.

$$A + B = C$$

$$A - B = C$$

DMI:	(C, A, NRA, DET)
TITLE:	MATRIX INVERSION/IN DOUBLE PRECISION
LANGUAGE:	FORTRAN
PARAMETERS:	
C	- INPUT MATRIX
A	- OUTPUT MATRIX
NRA	- NUMBER OF ROWS AND COLUMNS IN INPUT MATRIX
DET	- DETERMINANT OF C
SUBROUTINES:	DMINV
USAGE:	COMPUTES DOUBLE PRECISION INVERSE OF A MATRIX

DMM	(A, B, C, NRA, NCA, NRB, NCB)
TITLE:	DOUBLE-PRECISION MATRIX MULTIPLICATION
LANGUAGE:	FORTRAN

PARAMETERS:

A	-	MULTIPLIER MATRIX
B	-	MULTIPLICAND MATRIX
C	-	PRODUCT
NRA	-	NUMBER OF ROWS IN A
NCA	-	NUMBER OF COLUMNS IN A
NRB	-	NUMBER OF ROWS IN B
NCB	-	NUMBER OF COLUMNS IN B

SUBROUTINES:	NONE
USAGE:	DOUBLE PRECISION MATRIX MULTIPLIER.
	$A \times B = C$

DMSTR: (A, R, N, MSA, MSR)

TITLE: CHANGING STORAGE MODE OF A MATRIX

LANGUAGE: FORTRAN

PARAMETERS:

SAME AS MSTR IN SCIENTIFIC SUBROUTINE PACKAGE

USAGE: SAME AS MSTR EXCEPT DMSTR IS DOUBLE PRECISION (THIS SUBROUTINE IS CALLED BY THE DOUBLE-PRECISION VERSION OF EIGEN)

DMT	(A, AT,NRA, NCA)	
TITLE:	DOUBLE-PRECISION MATRIX TRANSPOSE	
LANGUAGE:	FORTRAN	
PARAMETERS:		
A	-	INPUT MATRIX OF DIMENSION (NRA X NCA)
AT	-	OUTPUT " " " (NCA X NRA)
NRA	-	NUMBER OF ROWS OF A
NCA	-	NUMBER OF COLUMNS OF A
SUBROUTINES:	NONE	
USAGE:	THIS SUBROUTINE GENERATES THE TRANSPOSE OF A MATRIX USING DOUBLE-PRECISION ARITHEMETIC.	

DPKCOV:	(DCS, R, ISB)	
TITLE:	PICTURE COVARIANCE MATRIX (MARKOV MODEL)	
LANGUAGE:	FORTRAN	
PARAMETERS:		
DCS	-	REAL COVARIANCE MATRIX OF DIMENSION ISB x ISB.
R	-	CORRELATION COEFFICIENT FOR A FIRST ORDER MARKOV PROCESS
ISB	-	NUMBER OF ROWS AND COLUMNS OF THE COVARIANCE MATRIX
SUBROUTINES	-	NONE
USAGE:	-	CONSTRUCTS THE COVARIANCE MATRIX (TOEPLITZ FORM) FOR A FIRST ORDER MARKOV PROCESS, USING DOUBLE PRECISION ARITHMETIC.

DPKNOS:	(DCN, SNR, ISL)	
TITLE:	WHITE-NOISE COVARIANCE MATRIX	
LANGUAGE:	FORTRAN	
PARAMETERS:		
DCN	-	NOISE COVARIANCE MATRIX OF DIMENSION (ESL X ISL)
SNR	-	SIGNAL TO NOISE RATIO
ISL	-	NUMBER OF ROWS AND COLUMNS OF DCN
SUBROUTINES:	-	NONE
USAGE:	-	CONSTRUCTS THE COVARIANCE MATRIX FOR WHITE-NOISE.

EXPAND (ARRAY, DIM1, DIM2)

DATE:	October 31, 1972
TITLE:	EXPAND PACKED DATA TO REAL *4 DATA
LANGUAGE:	ASSEMBLER

PARAMETERS:

ARRAY	-	NAME OF INPUT ARRAY
DIM1	-	ARRAY DIMENSION 1
DIM2	-	ARRAY DIMENSION 2

SUBROUTINES: NONE

USAGE: EXPANDS PACKED INTEGER DATA IN AND ARRAY INTO 4 BYTE REAL DATA. RESULTS ARE PLACED IN THE INPUT ARRAY. (INPUT ARRAY MUST BE LARGE ENOUGH TO HOLD RESULTS)

SEE IXPAND FOR FURTHER DETAILS.

FASTHAD

DATE: October 31, 1972

TITLE: ONE DIMENSIONAL HADAMARD TRANSFORM

LANGUAGE: ASSEMBLER

PARAMETERS:

CALL PROGRAM WITH:

```
CALL PR 256 (VIN, VOUT)
VIN  -  INPUT VECTOR (MUST BE DIMENSIONED 256)
VOUT  -  OUTPUT VECTOR (MUST BE DIMENSIONED 256)
```

FCOST: (ISIZE, ILINES, INDS, OUTDS, ITYPE)

TITLE: FAST COSINE TRANSFORM

LANGUAGE: FORTRAN

PARAMETERS:

ISIZE	:	NUMBER OF PICTURE ELEMENTS IN ONE LINE
ILINES	:	NUMBER OF LINES
INDS	:	INPUT DATA SET NUMBER (SYSUNIT #)
OUTDS	:	OUTPUT DATA SET NUMBER (SYSUNIT #)
ITYPE	:	TYPE OF THE TRANSFORM

ITYPE = +1	:	FORWARD FAST COSINE TRANSFORM
ITYPE=-1	:	INVERSE FAST COSINE TRANSFORM

SUBROUTINE: HARM

USAGE: THIS SUBROUTINE COMPUTES THE FORWARD OR INVERSE COSINE TRANSFORM OF A PICTURE USING THE FAST FOURIER TRANSFORM HARM.

FG256	(IRDS, ICDS, IORDS, IOCDS, NR, NC, JOPT)
DATE:	JUNE 11, 1973
TITLE:	GENERALIZED 2-DIM. FOURIER TRANSFORM
LANGUAGE:	FORTRAN

PARAMETERS:

IRDS	-	INPUT/OUTPUT DATA SET (REAL)
ICDS	-	INPUT/OUTPUT DATA SET (IMAGINARY)
IORDS	-	WORK DATA SET (REAL)
IOCDS	-	WORK DATA SET (COMPLEX)
NR	-	NUMBER OF ROWS OF INPUT ARRAY (MUST BE A POWER OF 2)
NC	-	NUMBER OF COLUMNS OF INPUT ARRAY (MUST BE A POWER OF 2)
JOPT(JOB OPTION)	-	1) NORMAL FOURIER TRANSFORM 2) DIRECT PRODUCT FOURIER TRANSFORM

SUBROUTINES: HARM, POWER

USAGE: THIS SUBROUTINE COMPUTES THE GENERALIZED TWO-DIMENSIONAL DISCRETE FOURIER TRANSFORM OF A COMPLEX ARRAY OF MAXIMUM SIZE 256 x 256. THE FUNCTION OF THIS SUBROUTINE IS THE SAME AS SUBROUTINE FORGEN WHICH WORKS FOR SMALLER ARRAYS.

FORGEN	(CR, CI, CFR, CFI, MC, NC, IR, IC, JOPT)	
TITLE:	GENERALIZED 2-DIM FOURIER TRANSFORM	
LANGUAGE:	FORTRAN	
PARAMETERS:		
CR	-	REAL PART OF INPUT MATRIX
CI	-	IMG. PART OF INPUT MATRIX
CFR	-	REAL PART OF OUTPUT MATRIX
CFI	-	IMG. PART OF OUTPUT MATRIX
MC	-	NUMBER OF ROWS OF C, MUST BE A POWER OF 2
NC	-	NUMBER OF COLUMNS OF C, MUST BE A POWER OF 2
IR=1,	FORWARD ROW TRANSFORM	
IR= -1	INVERSE ROW TRANSFORM	
IC=1,	FORWARD COLUMN TRANSFORM	
IC= -1,	INVERSE COLUMN TRANSFORM	
JOPT=1--	NORMAL FOURIER TRANSFORM	
JOPT=2--	DIRECT PRODUCT FOURIER TRANSFORM	
SUBROUTINES:	HARM, POWER	
USAGE:	THIS SUBROUTINE COMPUTES THE GENERALIZED TWO-DIMENSIONAL DISCRETE FOURIER TRANSFORM OF A COMPLEX ARRAY, THE FUNCTION OF THIS SUBROUTINE IS THE SAME AS SUBROUTINE FG256 WHICH WORKS FOR LARGER ARRAYS	

FORMAG:	(XR, XI, XM, M, N)
DATE:	JUNE 11, 1973
TITLE:	MAGNITUDE MATRIX OF A COMPLEX MATRIX
LANGUAGE:	FORTRAN
PARAMETERS:	
XR:	MATRIX OF REAL COMPONENTS
XI:	MATRIX OF IMAGINARY COMPONENTS
XM:	MATRIX OF MAGNITUDE COMPONENTS
M:	NUMBER OF ROWS
N:	NUMBER OF COLUMNS
SUBROUTINES:	NONE
USAGE:	THIS SUBROUTINE COMPUTES THE MAGNITUDES OF THE COMPONENTS OF A COMPLEX MATRIX.

HAAR(X, N)

TITLE: FAST FORWARD HAAR TRANSFORM

LANGUAGE: FORTRAN

PARAMETERS:

X	--	input vector; also transformed output vector.
N	--	length of vector X; must equal a power of 2.

SUBROUTINES; NONE

USAGE: This subroutine computes the fast forward Haar transform of any vector whose length is a power of two.

HAARGN:	(CIN, COUT, MC, NC, IR, IC)
TITLE:	GENERALIZED 2-DIM HAAR TRANSFORM
LANGUAGE:	FORTRAN

PARAMETERS:

CIN	-	INPUT MATRIX
COUT	-	OUTPUT MATRIX
MC	-	NUMBER OF ROWS OF C
NC	-	NUMBER OF COLUMNS OF C
IR=1,		FORWARD ROW TRANSFORM
IR=-1,		INVERSE ROW TRANSFORM
IC=1,		FORWARD COLUMN TRANSFORM
IC=-1,		INVERSE COLUMN TRANSFORM

SUBROUTINES:	HAAR, HAARI
USAGE:	THIS SUBROUTINE COMPUTES THE GENERALIZED 2-DIMENSIONAL HAAR TRANSFORM OF A REAL MATRIX.

HAARI (X,N)

TITLE: FAST INVERSE HAAR TRANSFORM

LANGUAGE: FORTRAN

PARAMETERS:

X	--	input vector; also inverse transformed output vector
N	--	length of vector X; must equal a power of 2.

SUBROUTINES: NONE

USAGE: This subroutine computes a fast inverse HAAR transform of any vector with length equal to a power of two.

HADGEN (CIN, COUT, MC, NC)

TITLE:	TWO-DIMENSIONAL HADAMARD TRANSFORM	
LANGUAGE:	FORTRAN	
PARAMETERS:		
CIN	-	INPUT MATRIX
COUT	-	OUTPUT MATRIX
MC	-	NUMBER OF ROWS OF C
NC	-	NUMBER OF COLUMNS OF C
SUBROUTINES:	IDHAD	
USAGE:	THIS SUBROUTINE COMPUTES THE TWO-DIMENSIONAL HADAMARD TRANSFORM OF A MATRIX.	

HISTR (IDS, ISIZE, AMAX, AMIN)

DATE:	October 31, 1972
TITLE:	HISTOGRAM
LANGUAGE:	FORTRAN

PARAMETER:

IDS	- INPUT DATA SET (SYS UNIT NUMBER + 1)
ISIZE	- # OF LINES IN PICTURE
AMAX	- LARGEST PICTURE VALUE
AMIN	- SMALLEST PICTURE VALUE

SUBROUTINES:

DSCIO

USAGE: PLOTS A HISTOGRAM OF THE NUMBER OF ELEMENTS AT EACH OF THE GREY LEVELS.

HNDEK (XR, XG, XB, C, M, Y)

TITLE: H AND D. CURVES HIGH SPEED EKTACHROME

LANGUAGE: FORTRAN

PARAMETERS:

XR	-	Red Layer Exposure
XG	-	Green Layer Exposure
XB	-	Blue layer exposure
C	-	Cyan Density Coefficient
M	-	Magenta Density Coefficient
Y	-	Yellow Density Coefficient

SUBROUTINES: DSKIO

USAGE: Input-XR, XG, XB obtained from subroutine XPOSEK

Output- C, M, Y are amounts of layer dyes resulting from given exposures.

Transmissivity of High Speed Ektachrome Transparency is given by:

$$T(\lambda) = 10^{-[C\text{-}DNC(\lambda)+M.DNM(\lambda)+Y.DNY(\lambda)]}$$

where DNC, DNM, DNY are spectral densities of unit amounts of layer dyes.

H16B (DI, DO)

DATE: October 31, 1972

TITLE: TWO DIMENSIONAL HADAMARD TRANSFORM IN 16 X 16 BLOCKS

LANGUAGE: FORTRAN

PARAMETER:

DI	-	INPUT MATRIX (16 x 16)
DO	-	OUTPUT MATRIX (16 x 16)

SUBROUTINES: PER16

USAGE:

IMPULS(H, BH, BV, IR, IC)

TITLE:	Gaussian Impulse Response Matrix
TYPE:	Subroutine
LANGUAGE:	FORTRAN
SUBROUTINES:	None
PARAMETERS:	H - Two dimensional impulse response matrix.
	IR - Number of rows in H (Must be odd integer).
	IC - Number of columns in H (must be odd integer).
	BH- Horizontal spread
	BV- Vertical spread
USAGE:	This subroutine constructs a Gaussian impulse response matrix.

I PRESS (ARRAY, DIM 1, DIM 2)

DATE:	October 31, 1972
TITLE:	INTEGER *4 TO PACKED INTEGER DATA
LANGUAGE:	ASSEMBLER

PAPRAMETERS

ARRAY	- NAME OF INPUT ARRAY
DIM 1	- ARRAY DIMENSION 1
DIM 2	- ARRAY DIMENSION 2

SUBROUTINES: NONE

USAGE: CONVERTS INTEGER *4 DATA TO PACKED INTEGER DATA. RESULTS ARE RETURNED THROUGH INPUT ARRAY.

SEE IXPAND FOR FURTHER DETAILS

IXPAND (ARRAY, DIM1, DIM2)

DATE:	October 31, 1972
TITLE:	PACKED INTEGER TO INTEGER *4
LANGUAGE:	ASSEMBLER

PARAMETERS:

ARRAY	- NAME OF INPUT ARRAY
DIM1	- ARRAY DIMENSION 1
DIM2	- ARRAY DIMENSION 2

SUBROUTINES: NONE

USAGE: CONVERTS PACKED INTEGER DATA TO INTEGER *4 DATA. RESULTS ARE PLACED IN THE INPUT ARRAY. (THE INPUT ARRAY MUST BE LARGE ENOUGH TO HOLD THE RESULTS)

```
DIMENSION IXYZ (16)
          .
          .
          .
CALL IPRESS (IXYZ, 16, 1)
CALL DSKIO(IXYZ, 16, 1, 0, 1)
          .
          .
          .
CALL DSKIO (IXYZ, 16, 1, 1, 1)
CALL IXPAND (IXYZ, 16, 1)
```

THIS EXAMPLE SIMPLY COMPRESSES DATA & STORES IT ON DISC IN PACKED FORMAT (NOTICE ONLY 25% AS MUCH DISC SPACE IS REQUIRED WHEN IXYZ IS PACKED). THE PROGRAM THEN READS IT FROM THE DISC AND EXPANDS IT TO A USEFUL FORTRAN FORMAT (INTEGER *4).

KL (RO, EV, EI)

TITLE: KARHUNEN-LOÈVE TRANSFORM

LANGUAGE: FORTRAN

PARAMETERS:

RO	-	CORRELATION COEFFICIENT OF FIRST-ORDER MARKOV PROCESS
EV	-	EIGEN VECTORS
EI	-	EIGEN VALUES

SUBROUTINES: NONE

USAGE: COMPUTES THE KARHUNEN-LOEVE TRANSFORM MATRIX FOR A FIRST-ORDER MARKOV PROCESS

MAS	(A, P, C, NR, NC, IAS)
TITLE:	MATRIX ADDITION AND SUBTRACTION
LANGUAGE:	FORTRAN
PARAMETERS:	SAME AS DMAS
SUBROUTINES:	NONE
USAGE:	SINGLE PRECISION MATRIX ADDITION AND SUBTRACTION
	$A + B = C$
	$A - B = C$

MAT 31 (N, R, G, B, U, V, W, UL, VL, WL)

DATE: October 31, 1972

TITLE: COLOR COORDINATE CONVERSION

LANGUAGE: FORTRAN

PARAMETERS:

N	-	3 x 3 COLOR COORDINATE CONVERSION MATRIX
R, G, B	-	TRISTIMULUS INPUT COMPONENTS
U, V, W	-	TRISTIMULUS OUTPUT COMPONENTS
UL, VL, WL	-	OUTPUT CHROMATICITY COORDINATES

SUBROUTINES: NONE

USAGE:

MI	(C, A, NRA, DET)	
TITLE:	MATRIX INVERSION	
LANGUAGE:	FORTRAN	
PARAMETERS:		
C	-	INPUT MATRIX
A	-	OUTPUT MATRIX
NRA	-	NUMBER OF ROWS AND COLUMNS IN INPUT MATRIX
DET	-	DETERMINANT OF C
SUBROUTINES:		MINV
USAGE:	COMPUTES INVERSE OF A MATRIX. (FOR DOUBLE PRECISION, SEE DMI)	

MM	(A, B, C, NRA, NRB, NCB)
TITLE:	MATRIX MULTIPLICATION
LANGUAGE:	FORTRAN
PARAMETERS:	SAME AS SUBROUTINE DMM
SUBROUTINES:	NONE
USAGE:	MATRIX MULTIPLICATION AB=C

MPLOT (A, B, C, IMIN, IMAX)

DATE:	October 31, 1972
TITLE:	VECTOR PLOT
LANGUAGE:	FORTRAN

PARAMETERS:

A	- REAL INPUT VECTOR
B	- REAL INPUT VECTOR
C	- REAL INPUT VECTOR
IMIN	- START ABCISSAS
IMAX	- FINISH ABCISSAS

SUBROUTINES: NONE

USAGE: PLOTS 3 VECTORS ON A COMMON SCALE. VECTORS A, B, C ARE REAL, BIPOLAR & DIMENSIONED UP TO 256. INPUT VECTORS ARE NOT MODIFIED.

A PLOTS	x
B PLOTS	*
C PLOTS	+

IN CASE OF OVERLAP C TAKES PRECEDENCE OVER B & B OVER A.

TO PLOT FEWER THAN THREE VECTORS, REPEAT VECTOR IN CALLING SEQUENCE, I. E.

```
CALL PLOT (A, A, A, IMIN, IMAX)
```

GETTING OUT (continued)

CDSPY:

1. Normal exit - Keep running program until asked for "WHICH DATA SETS". Respond by typing '0' (zero).

2. Abnormal - Strike key while picture is being displayed. If computer give back an "*", type 'OF, CDSPY, 1'. If it does not, then re-initialization may be necessary. Repeated difficulty with displaying may indicate hardware problem. See documentation for CDSPY.

MARK:

1. Normal exit - Turn on bit #5 of the computer console, and wait for the message "ENTER INCREMENT". Type the number zero.

2. Abnormal - Strike any alphanumeric key. If the computer gives back an "*", type 'OF, MARK, 1'. If it does not respond to the key, re-initialization of the system may be necessary (See front of this manual for "STARTING UP THE SYSTEM").

FSCAN:

This program is not to be run, except by Dr. Pratt or Mark Sanders.

1. Normal exit - Finish displaying the frames currently in progress, and wait for the question "DONE? ". Type 'YES', or any combination beginning with the letters 'YE'.

2. Abnormal exit - Halt the computer and re-initialize the system.

ALL OTHER PROGRAMS:

1. Normal exit - Wait until asked for the option to restart or exit (or, in some cases, a third option). Type the integer which corresponds to the respective position of the desired option within the question asked. In general, restart will be abbreviated "RE", and exit will be abbreviated "EX".

2. Abnormal exit - Halt the computer and re-initialize. See front section of this manual for "STARTING UP THE SYSTEM".

FINAL IMPORTANT NOTE:

Most programs terminate by typing "XXXXX: STOP", where XXXXX is the program name. However, some programs, such as ASCAN and CDSPY stop without printing a message. The best way to determine if a program has terminated is to attempt to execute a new command. If the computer processes the command, all is well. If not, look for the problems mentioned in the various sections of this manual.

LOGICAL UNIT NUMBERS

1	Teletype #1
2	System Disc
3	Nothing
4	Paper Tape Punch
5	Paper Tape Reader
6	Line Printer
7	7-Track Tape Unit
8	Dummy Driver Channel
9	9-Track Tape Unit (subchannel 0)
10	9-Track Tape Unit (subchannel 1)
11	Data Disc (fixed pack)
12	Data Disc (removable pack)
13	Teletype #2
14	Card Reader

EQUIPMENT TABLE

1 -	Line Printer
2 -	Paper Tape Punch
3 -	Paper Tape Reader
4 -	Teletype
6 -	System Disc
7 -	Tape Unit
9 -	Data Disc

DEMONSTRATION PACK DIRECTORY

Data set	Description
19	Kodak standing couple, red field
20	Kodak standing couple, green field
21	Kodak standing couple, blue field
22	2-bit slant threshold coded girl (STCG)
23	1.5-bit STCG
24	1.0-bit STCG
25	Kodak girl, red field
26	Kodak girl, green filed
27	Kodak girl, blue field
28	Black & white girl
29	Black & white couple
30	Black & white moonscene
31	Original tank } 6-bit
32	Noisy tank } 6-bit
33	Corrected tank } 6-bit
34	Circular blurred moon
35	.5-bit STCG
36	1.5-bit slant zonal coded girl

CONTENTS OF DEMO PACK #2

Data set	Description	
19.	red	
20.	green	#1871 2-bit slant color girl
21.	blue	
22.	red	
23.	green	#1761 3-bit slant color girl
24.	blue	
25.		
26.		
27.	Resolution chart	
28.	red	
29.	green	Mandrill (8-bit)
30.	blue	
31.	Frog (6-bit)	
32.	Paper tape subtraction (8-bit)	
33.	Bright paper tape (8-bit)	
34.	Pseudo/true color ape (6-bit)--use chart 14	
35.	Mark's baby picture	
36.	Sandy	

MAG TAPE FILES FILES FROM WHICH KRUGER
DEMO DISK PACK WAS MADE

Disk Pack D. S.

19	PHD001	35
20	"	12
21	"	36
22	"	10
23	"	31
24	"	32
25	"	33
26	"	34
27	2018X	9
28	"	10
29	"	11
30	ANNE01	1
31	"	2
32	"	4
33	CMK001	1
34	"	2
35	RPK001	9
36	"	10

PSEUDO CHARTS

1. Harry Andrews' chart
2. Smooth colors
3. Primary shades
4. Bright, sharp transitions
5. Pastel contrast
6. Three color blocks
7. Pastels only
8. C. D. Taylor colors
9. 16-level grey scale (4 bits)
10. Grey, color alternation
11. Half grey, half color
12. 8-level grey scale (3 bits)
13. 4-level grey scale
14.
15.
16.
17. Rich smooth transitions
18. All black except location 1.
19.
20.
21.
22.
23.
24.
25.

CD Numbers

1.	Full 4x4	-1
2.	Full 2x2	0
3.	X checker	-27031
4.	O checker	27030
5.	(full)-corners	28662
6.	Full diagonal	13186
7.	Small 2x2	1632
8.	Full window	-1633
9.	Small x	1317
10.	Small diamond (plus signs)	626
11.	Small oval	854
12.	Horizontal 3x2	119
13.	Horizontal 4x2	255
14.	Vertical 4x2	13107
15.	Vertical 3x2	819
16.	Fine mesh	23130
17.	1x2 Strip mesh	1686
18.	3x3 block	1911

6. BASIC BINARY LOADER (BBL) - BASIC BINARY DISC LOADER (BBDL)

BBDL

memory address		instruction	memory address		instruction
077700	=	107700	077740	=	102055
077701	=	002401	077741	=	027700
077702	=	063726	077742	=	000000
077703	=	006700	077743	=	006600
077704	=	017742	077744	=	103713
077705	=	007306	077745	=	102313
077706	=	027713	077746	=	027745
077707	=	002006	077747	=	107413
077710	=	027703	077750	=	002041
077711	=	102077	077751	=	127742
077712	=	027700	077752	=	005767
077713	=	077754	077753	=	027744
077714	=	017742	077754	=	000000
077715	=	017742	077755	=	100100
077716	=	074000	077756	=	020016
077717	=	077757	077757	=	000000
077720	=	067757	077760	=	107700
077721	=	047755	077761	=	063756
077722	=	002040	077762	=	102606
077723	=	027740	077763	=	002700
077724	=	017742	077764	=	102717
077725	=	040001	077765	=	001500
077726	=	177757	077766	=	102602
077727	=	037757	077767	=	063777
077730	=	000040	077770	=	102702
077731	=	037754	077771	=	102602
077732	=	027720	077772	=	103706
077733	=	017742	077773	=	102716
077734	=	054000	077774	=	067776
077735	=	027702	077775	=	074077
077736	=	102011	077776	=	024077
077737	=	027700	077777	=	177700

MV: (A, B, C, MC, NC)

TITLE:	MATRIX-VECTOR MULTIPLICATION	
PARAMETERS:		
A	-	INPUT MATRIX
B	-	INPUT VECTOR
C	-	OUTPUT VECTOR
MC	-	NUMBER OF ROWS OF A
NC	-	NUMBER OF COLUMNS OF A
SUBROUTINES:		NONE
USAGE:		THIS SUBROUTINE PERFORMS A MATRIX-VECTOR MULTIPLICATION.

PERM:	(K, N)	
TITLE:	PERMUTATION FOR HADAMARD TRANSFORM	
PARAMETERS:		
K	-	INPUT VECTOR
N	-	SIZE OF THE INPUT VECTOR
SUBROUTINES:	NONE	
USAGE:	THIS SUBROUTINE FILLS THE INPUT VECTOR K WITH INTEGERS 1 THROUGH N IN SUCH ORDER THAT SAMPLES IN THE HADAMARD TRANSFORM MAY BE REARRANGED AS TO SEQUENCY	

PR256(VIN, VOUT)

DATE:	October 31, 1972
TITLE:	ONE DIMENSIONAL HADAMARD TRANSFORM
LANGUAGE:	ASSEMBLER
PARAMETERS:	
VIN -	INPUT VECTOR (MUST BE DIMENSIONED 256)
VOUT -	OUTPUT VECTOR (MUST BE DIMENSIONED 256)

PIKCOV	(DCS, R, ISB)
TITLE:	PICTURE COVARIANCE MATRIX (MARKOV MODEL)
LANGUAGE:	FORTRAN
PARAMETERS:	SAME AS DPIKCOV
USAGE:	SAME AS DPIKCOV, EXCEPT PIKCOV USES SINGLE PRECISION ARITHMETIC.

PIKNOS	(DCN, SNR, ISL)
TITLE:	WHITE-NOISE COVARIANCE MATRIX
LANGUAGE:	FORTRAN
PARAMETERS:	SAME AS DPIKNOS
USAGE:	SAME AS DPIKNOS, EXCEPT THAT PIKNOS USES SINGLE PRECISION ARITHMETIC.

PIKOUT (NDS1, NDS2, NDS3, ISIZE, CLIP, DOMAIN, NPIK, JOPT, NBIT)

DATE:	FEBRUARY 7, 1973
TITLE:	BLACK & WHITE PICTURE OUTPUT
LANGUAGE:	FORTRAN
SUBROUTINES:	DSKIO

USAGE:

PICTURE OUTPUT PROGRAM
PERFORMS HISTOGRAMS
PERFORMS SCALING
PERFORMS CONVERSION TO BYTE FORMAT
FRAMES PIK WITH GRAY SCALE AND LABEL NUMBER

PIKOUT IS CALLED FROM A FORTRAN PROGRAM AS FOLLOWS:

```
        CALL PIKOUT(NDS1,NDS2,NDS3,ISIZE, CLIP,DOMAIN,NPIK,JOPT,NBIT)
WHERE
        CLIP IS AN ARRAY OF 2 REAL NUMBERS:     CLIP(1) = MAX CLIP
                                                CLIP(2) = MIN CLIP
        DOMAIN IS AN ARRAY OF 2 REAL NUMBERS:   DOMAIN(1) = MAX DOMAIN
                                                DOMAIN(2) = MIN DOMAIN
        NPIK IS AN ARRAY OF 4 INTEGERS FOR THE FRAME EXPERIMENT
             NUMBER AT THE BOTTOM OF THE NEW PIK
        ISIZE IS THE SIZE OF THE PIK (NUMBER OF LINES)
        NBIT IS THE NUMBER OF BITS PER PIXEL
        NDS1, NDS2, NDS3 ARE DATA SETS
             NDS1 IS A REAL DATA SET USED FOR INPUT FOR OPTIONS 1
                THROUGH 7
             NDS2 IS A REAL DATA SET USED FOR OUTPUT FOR OPTIONS 1
                THROUGH 7 AND FOR INPUT FOR OPTIONS 8-9
             NDS3 IS A PACKED DATA SET USED FOR OUTPUT FOR OPTIONS
                3, 4, 6, 7, 8, 9, AND BOTH INPUT AND OUTPUT FOR OPTION
                0
```

OPTIONS ARE		NDS1	NDS2	NDS3
0	FRAME THE PACKED PIK ON NDS3			I, 0
1	HISTOGRAM	I	0	
2	HISTOGRAM, SCALING	I	0	
3	HISTOGRAM, SCALING, PACKING	I	0	0
4	HISTOGRAM, SCALING, FRAMING, PACKING	I	0	0
5	SCALING		I, 0	
6	SCALING, PACKING		I, 0	0
7	SCALING, FRAMING, PACKING		I, 0	0
8	FRAMING, PACKING		I, 0	0
9	PACKING		I	0

*I = USES THIS DATA SET FOR INPUT, 0 = USES DATA SET FOR OUTPUT

SCALING IMPLIES A HISTOGRAM OF THE SCALED PIK
IF CLIP(1) AND CLIP(2) ARE BOTH ZERO, PIK WILL BE CLIPPED AT
ACTUAL MAXIMUM AND MINIMUM (IF SCALING SPECIFIED)

POWER	(JJ, I POWER)
TITLE:	LOG-2 OF AN INTEGER
LANGUAGE:	FORTRAN
PARAMETERS:	
JJ	- INPUT INTEGER = 2 ** JJ
IPOWER	- $LOG_2(JJ)$
SUBROUTINES:	NONE
USAGE:	THIS SUBROUTINE COMPUTES THE LOGARITHM (OF BASE 2) OF AN INTEGER WHICH IS A POWER OF 2, BY SUCCESSIVE DIVISIONS BY 2.

PRESS (ARRAY, DIM1, DIM2)

DATE:	October 31, 1972
TITLE:	REAL *4 TO PACKED INTEGER
LANGUAGE:	FORTRAN

PARAMETER:

ARRAY	-	NAME OF INPUT ARRAY
DIM1	-	ARRAY DIMENSION 1
DIM2	-	ARRAY DIMENSION 2

SUBROUTINES: NONE

USAGE: CONVERTS REAL *4 DATA TO PACKED INTEGER DATA. RESULTS ARE PLACED IN THE INPUT ARRAY.

SEE IXPAND FOR FURTHER DETAILS.

SIMEQ (A, SRU, SRV, SRW, X, Y, Z)

TITLE: SIMULTANEOUS EQUATIONS

LANGUAGE: FORTRAN

PARAMETERS:

A	-	Given 3 X 3 Matrix, A(3, 3)
SRU	-	Given Vector
SRV	-	
SRW	-	
X	-	Solution Vector
Y	-	
Z	-	

SUBROUTINES: NONE

USAGE: Solves Matrix equation

$$\begin{pmatrix} SRU \\ SRV \\ SRW \end{pmatrix} = \begin{pmatrix} A \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

FOR X, Y, Z

SLAGEN (CIN, COUT, MC, NC, IT, IC)

TITLE:	GENERALIZED 2-DIM. SLANT TRANSFORM
LANGUAGE:	FORTRAN

PARAMETERS:

CIN	-	INPUT MATRIX
COUT	-	OUTPUT MATRIX
MC	-	NUMBER OF ROWS OF C
NC	-	NUMBER OF COLUMNS OF C
IR=1,	-	FORWARD ROW TRANSFORM
IR=-1,	-	INVERSE ROW TRANSFORM
IC=1,	-	FORWARD COLUMN TRANSFORM
IC=-1,	-	INVERSE COLUMN TRANSFORM

SUBROUTINES:	SLANT, SLANTI
USAGE:	THIS SUBROUTINE COMPUTES THE GENERALIZED 2-DIMENSIONAL SLANT TRANSFORM OF A REAL MATRIX.

SLANT (X, N)

TITLE: FAST FORWARD SLANT TRANSFORM

LANGUAGE: FORTRAN

PARAMETERS:

X	--	Input vector; also transformed output vector.
N	--	length of vector X; must equal a power of 2.

SUBROUTINES: NONE

USAGE: This subroutine computes the fast forward slant transform of any vector of length equal to a power of two.

SLANTI	(X, N)
TITLE:	FAST INVERSE SLANT TRANSFORM
LANGUAGE:	FORTRAN
PARAMETERS:	
X --	input vector; also inverse transformed output vector
N --	length of vector X; must equal a power of 2.
SUBROUTINES:	NONE
USAGE:	This subroutine computes the fast inverse slant transform of any vector of length equal to a power of two.

TRNSCO (PTYPE, RO, RESULT)

TITLE:	COVARIANCE MATRIX OF THE TRANSFORM OF A VECTOR
LANGUAGE:	FORTRAN

PARAMETERS:

PTYPE -	TYPE OF THE UNITARY TRANSFORMATION
RO -	CORRELATION COEFFICIENT OF THE FIRST ORDER MARKOV PROCESS.
RESULT -	COVARIANCE MATRIX OF THE TRANSFORM OF THE FIRST ORDER MARKOV PROCESS.
SUBROUTINES:	PIKCOV, FORGEN, FORMAG, HADGEN, SLAGEN, COSIGN, KL, HAARGN
USAGE:	THIS SUBROUTINE TAKES THE COVARIANCE MATRIX OF THE FIRST ORDER MARKOV PROCESS (WITH COEFFICIENT RO) AND COMPUTES THE COVARIANCE MATRIX OF THE TRANSFORM. THE THE TRANSFORMS CONSIDERED ARE FOURIER, HADAMARD, SLANT, COSINE, KARHUNEN-LOEVE AND HAAR.

UVWC (COL, U, V, W, UL, VL, WL)

TITLE: UCS Tristimulus and chromaticity

LANGUAGE: FORTRAN

PARAMETERS:

COL	-	Color spectral intensity Col(I), I=1,80 (wavelength = 375+5*I nanometers)
U	-	
V	-	UCS (uniform Chromaticity Scale) Tristimulus values of col
W	-	
UL	-	
VL	-	UCS Chromaticities of col
WL	-	

SUBROUTINES: DSKIO

USAGE: Input - Col (I) I=1,80.

Output - U,V,W,VL,VL,WL.

U,V,W are normalized by luminance of reference white (Illuminant C).

XPOSEK (COL, XR, XG, XB)

TITLE: LAYER EXPOSURES OF HIGH SPEED EKTACHROME

LANGUAGE: FORTRAN

PARAMETERS:

COL	-	Spectral intensity of color, Col(I), I=1,80 Wavelength = 375+5*I NM.
XR	-	Red exposure
XG	-	Green exposure
XB	-	Blue Exposure

SUBROUTINES: DSKIO

USAGE: Computes exposures of color film layers for high speed ektachrome. Exposures are normalized to give values of 100 for reference white (Illuminant C).

III. ECL ABSOLUTE PHASE LIBRARY

ABSOLUTE PHASE LIBRARY

The following programs are stored on the IBM 360/44 Phase Library (IPLABS). They may be executed with the statement:

```
//name EXEC phase name(parameter, parameter,...)
```

Assuming IPLABS has been accesseu to SYSRUN in a previous job statement:

```
//SYSRUN ACCESS IPLABS
```

Input and output from these phases are on the data set numbers specified under each individual phase and the parameter list is needed only if the phase requires parameters.

CONTRAST

DATE: October 31, 1972

TITLE: SCALLING PROGRAM

TYPE: Phase

PARAMETERS: none

USAGE: This phase reads REAL*4 data from disk on SYS001 & scales it to 64 levels (0-63). Output is on SYS005.

DSKIO (VECT, IBYTE, ILINE, IRDRW, IDS)

DATE: October 31, 1972

TITLE: DISK INPUT & OUTPUT

TYPE: ASSEMBLER SUBROUTINE

PARAMETERS:

VECT	Name of Data Vector
IBYTE	Number of bytes per line
ILINE	Line number
IRDRW	1 = read; 0 = write
IDS	Data set number

USAGE: This subroutine may be used in any FORTRAN program as follows to read 512 lines of length 256 bytes from test.

```
DO 1  I = 1,512
CALL DSKIO(TEST,256,I,1,2,)
```

See examples

MULTIPLX

DATE:	February 8, 1973
TITLE:	AEROJET FORMATER
USAGE:	This phase reads 3 packed data sets and combines them into one packed Aerojet formated data set.

INPUT DATA SETS:
SYS001 - Blue packed data set
SYS002 - Red packed data set
SYS003 - Green packed data set

Output data set is on SYS004.

NEWINPUT

DATE: October 31, 1972

TITLE: PACKED DATA TO REAL*4 DATA

TYPE: Phase

PARAMETERS: none

USAGE: This phase converts packed integer data to REAL*4 data. Input data is on SYS005 and the output data is on SYS001.

See examples.

PIKOUT

DATE:	November 20, 1972
TITLE:	B & W PICTURE OUTPUT
TYPE:	Phase
PARAMETERS:	None
SUBROUTINES:	PIKOUT, DSKIO
USAGE:	

PROGRAM OPTION AND I/O TABLE

OPTION	DESCRIPTION	NDS1	NDS2	NDS3
0	Frame the packed picture			I, 0
1	Histogram	I	0	
2	Histogram, Scaling	I	0	
3	Histogram, Scaling, Packing	I	0	0
4	Histogram, Scaling, Packing, Framing	I	0	0
5	Scaling		I, 0	
6	Scaling, Packing		I, 0	0
7	Scaling, Packing, Framing		I, 0	0
8	Framing, Packing		I, 0	0
9	Packing		I	0

I = uses this data set for input

0 = uses this data set for output

The program parameters are read in with 4 data cards.

DATA CARD 1

NDS1, NDS2, NDS3

DATA CARD 2

MAXCLIP MINCLIP

DATA CARD 3

MAXDOMAIN MINDOMAIN

DATA CARD 4

JOPT, BITS, EXPMT, LINES

PIKOUT

Page 2 of 2

NDS1 - Real data set numoer (I1 Format)

NDS2 - Real data set number (I1)

NDS3 - Packed data set number (I1)

MAXCLIP	-	MAXIMUM CLIP VALUE (must be real number with decimal point, F10. 5 Format)
MINCLIP	-	MINIMUM CLIP VALUE (must be real number with decimal point and start in column 11) No comma should separate MAXCLIP and MINCLIP. (F10. 5 Format)
MAXDOMAIN	-	MAXIMUM DOMAIN VALUE (Same format as MAXCLIP)
MINDOMAIN	-	MINIMUM DOMAIN VALUE (Same format as MINCLIP)
JOPT #	-	PROGRAM OPTION NUMBER (I1 Format)
# BITS	-	NUMBER OF BITS PER PIXEL (I1 Format)
EXPMT #	-	FOUR INTEFERS GIVING EXPERIMENT NUMBER (I4 Format)
# LINES	-	NUMBER OF LINES IN PICTURE (I3 Format)

If MINCLIP = MAXCLIP then the program will automatically CLIP at the actual MAX and MIN of the picture

EXAMPLE:

```
// EXEC PIKOUT
1,2,3
0.0          0.0
255.0        0.0
4,8,0001,256
```

SQUARE

DATE: October 31, 1972

TITLE: MATRIX TRANSPOSE

TYPE: Phase

PARAMETERS: None

USAGE: This phase transposes a square matrix of REAL*4 numbers. Input is on SYS002 & output is on SYS001.

STAT (INPUT OPTION, OUTPUT OPTION(S))

DATE:	November 20, 1972
TITLE:	STATISTICAL ANALYSIS OF PICTURES
TYPE:	Phase
PARAMETERS:	Any number of output options up to 5 may be used.

A. INPUT OPTIONS:

INTEGER	-	INPUT IS AN INTEGER PIK
REAL	-	INPUT IS A REAL PIK
EXPAND	-	INPUT IS "PACKED," WILL EXPAND TO REAL PIK (WILL USE SYS004)
IXPAND	-	INPUT IS "PACKED," WILL IXPAND TO INTEGER PIK (WILL USE SYS004)
		(ALL INPUT IS ON SYS002)

B. PROGRAM OPTIONS:

HISTOGRM	-	DISPLAY A HISTOGRAM (MEAN, VARIANCE & ZEORTH ORDER ENTROPY ALSO GIVEN)
LCOVAR	-	DISPLAY LINE COVARIANCE
CCOVAR	-	DISPLAY COLUMN COVARIANCE (NEEDS TURNED IMAGE ON SYS003 EITHER SUPPLIED OR CREATED BY TURN)
TURN	-	CREATE A 90° ROTATED PIK ON SYS003. (WILL USE SYS005)
LFOE	-	DISPLAY LINE FIRST ORDER ENTROPY & HISTOGRAM (WILL USE SYS005 & SYS006)
CFOE	-	DISPLAY COLUMN FIRST ORDER ENTROPY & HISTOGRAM (NEEDS TURNED IMAGE ON SYS003 & WORK AREA ON SYS005 & SYS006)
PRINT	-	PRINT LINES 1-10, 124-134, & 247-256.
BFOE	-	BOTH LFOE & CFOE

C. STANDARD SYSUNIT ASSIGNMENTS:

SYS002	-	SHOULD BE 256 BYTES BY 256 LINES FOR PACKED PIK AND 1024 BYTES BY 256 LINES FOR REAL OR INTEGER PIK.

SYS003 through SYS006 - SHOULD BE 1024 BYTES BY 256 LINES

THIS PHASE REQUIRES 64K TO 72K OF CORE.

EXAMPLE:

```
// EXEC STAT(REAL,LCOVAR,TURN,CCOVAR,BFOE,PRINT)
```

UNPACKI

DATE:	February 7, 1973
TITLE:	UNPACK TO INTEGER*4
TYPE:	Phase
PARAMETERS:	None
USAGE:	This phase reads packed data from SYS002, unpacks it to INTEGER and writes results on SYS003.

UNPACKR

DATE:	February 7, 1973
TITLE:	UNPACK TO REAL*4
TYPE:	Phase
PARAMETERS:	None
USAGE:	This phase reads packed data from SYS002, unpacks it to REAL, and writes the real results on SYS003.

X

DATE:	October 31, 1972
TITLE:	END OF JOB STEP
TYPE:	PHASE
PARAMETERS:	NONE
USAGE:	THIS PROGRAM WILL END THE JOB STEP WHEN USED IN THE OPTION LIST OF A PHASE THAT TERMINATES BY LINKING TO ANOTHER PHASE (FOURIER AND HADAMARD TRANSFORM PHASES).

FOURIER TRANSFORM PHASES

The Fourier transform system consists of seven phases. Three phases (PASSONE, TRANSONE, PASSTWO) compute the forward transform, one phase does filtering (FILTERS) and the last three phases (PASSTHRE, TRANSTWO, PASSFOUR) take the reverse transform.

For all the Fourier systems there are the following standard unit # assignments and data set dimensions:

SYS UNIT #	# OF BLOCKS	BLOCKSIZE
SYS001	# of picture lines	4* number of blocks
SYS002	(# of picture lines/2)+ 1	8* number of picture lines
SYS004	(# of picture lines/2) + 1	8* number of picture lines
SYS005	# of picture lines	# of picture lines

To get the frequency domain (Forward Transform):

```
        // EXEC PASSCNE(PASSTWO)
```

This statement executes PASSONE, TRANSONE and PASSTWO. Input is on SYS001 while output is on SYS004.

To filter frequency domain and take reverse transform:

```
        // EXEC FILTERS(PASSFOUR,X)
         FILTERS DATA CARD 1
         FILTERS DATA CARD 2
```

This statement executes phase FILTERS(Input is on SYS004) and then PASSTHRE, TRANSTWO and finally PASSFOUR(Output is on SYS001). The X terminates the job step. In place of X any other phase could be inserted and it would then be executed.

To get reverse transform without FILTERING:

```
        // EXEC PASSTHRE(PASSFOUR,X)
```

This executes PASSTHRE, TRANSTWO and PASSFOUR. X then terminated the job step. Input is on SYS002 while output is on SYS001.

PASSONE (phase)

DATE: October 31, 1972

TITLE: 1-D FOURIER TRANSFORM (REAL*4)

TYPE: Phase

PARAMETERS: This phase links to 'TRANSONE' and subsequently to any phase listed in the parameter list.

USAGE: Performs one-dimensional Fourier transform on REAL*4 data. Resultant transform is COMPLEX*8. Input is on SYS001 and output is on SYS001 & SYS004.

TRANSONE (phase)

DATE: October 31, 1972

TITLE: TRANSPOSE MATRIX (complex *8)

TYPE: Phase

PARAMETERS: This Phase links to a phase specified in the parameter list.

USAGE: This transposes a matrix of complex numbers. Input is on SYS001 and output is on SYS004. This phase is intended for use after PASSONE.

PASSTWO

DATE: October 31, 1972

TITLE: 1-D FOURIER TRANSFORM (COMPLEX*8)

TYPE: Phase

PARAMETERS: none

USAGE: Performs line by line 1-D Fourier transform on COMPLEX*8 data with resultant transform COMPLEX*8. Input & output is on SYS004.

PASSTHRE (phase)

DATE: October 31, 1972

TITLE: 1-D INVERSE FOURIER TRANSFORM (COMPLEX*8)

TYPE: Phase

PARAMETERS: This phase links to 'TRANSTWO' and subsequently to any phase listed in the parameter list.

USAGE: Performs the inverse of 'PASSTWO'. Input & output is on SYS002.

TRANSTWO (phase)

DATE: October 31, 1972

TITLE: INVERSE OF TRANSONE

TYPE: Phase

PARAMETERS: This phase links to any phase specified in option list.

USAGE: This phase performs the inverse of 'TRANSONE'. It is intended for use after PASSTHRE.

PASSFOUR (phase)

DATE: October 31, 1972

TITLE: 1-D INVERSE FOURIER TRANSFORM

TYPE: Phase

PARAMETERS: This phase links to any phase specified in parameter list.

USAGE: Performs line by line 1-D inverse Fourier transform on Complex data with output REAL*4.

DATE:	October 31, 1972
TITLE:	FOURIER DOMAIN FILTERING
TYPE:	Phase
PARAMETERS:	This phase will link to specified phase in option list
USAGE:	The program performs FILTERING in the Fourier comples frequency domain as follows:

FILTER OPTIONS

1. Low Pass
2. High Pass
3. Band Pass
4. Gaussian
5. Inverse Gaussian
6. Bessel (J_1)
7. Inverse Bessel ($1/J_1$)
8. Log (Magnitude + 1) & retain original phase
9. Set magnitude to 1 & retain original phase
10. Nth root of magnitude & retain original phase

FILTERING options are input to phase through 2 data cards.

CARD 1

Columns (1-2)(I2 Format) Number of FILTERS to be used.

Columns (3-4, 5-6, ... 21-22)(I2, I2,...) FILTERS to be applied specified by option number. FILTERS will be processed in order they appear, with all nonmultiplicative FILTERS being processed first.

CARD 2

Columns (1-3)(I3 Format) Low frequency cutoff (expressed as radius in the frequency domain).

Columns (4-6)(I3) High frequency cutoff (expressed as radius in the frequency domain).

Columns (11-20)(F10. 3) Standard deviation for FILTERS based on Gaussian Distribution).

Columns (21-30)(F10. 3) Multiplies for FILTERS based on Bessel function.

Columns 31-40)(F10. 3) Root for ROOT FILTER.

COLUMN (80)(I1)	Stop parameter. If column 80 is blank, program links to PASSTHRE. If column 80 is 1, program terminates after filtering is complete. This can be useful for looking at power spectrum filters.

Input is on SYS004 and output is on SYS002.

HADAMARD TRANSFORM PHASES

STANDARD UNIT ASSIGNMENTS

UNIT	NO/BLKS	BLOCKSIZE (BYTES)
SYS001	Number of picture lines	4* number of blocks
SYS002	Same as SYS001	Same as SYS001
SYS004	Same as SYS001	Same as SYS001
SYS005	Same as SYS001	Number of picture lines

To get to Hadamard domain:

```
// EXEC WALSHER (#2, SQUARE, WALSHER, #1, X)
/*
```

To get from Hadamard to real domain with filtering

```
// EXEC REALFLTR (#2, SQUARE, WALSHER, #1, Option)
          data card 1
          data card 2
          (data card 3)
          /*
```

Where option is either CONTRAST, HISTEQ or X. Otherwise, same as filters for Fourier. Data cards 1 and 2 are for phase 'FILTERS' and data card 3 is for phase 'CONTRAST' and need only be included if CONTRAST is specified.

REALFLTR

DATE: October 31, 1972

TITLE: REAL FILTER

TYPE: Phase

PARAMETERS: SAME AS FILTERS

SAME FILTERING OPTIONS AS 'FILTERS'

USAGE: This phase performs filtering in REAL*4 Hadamard domain. Input is on SYS004 & output is on SYS002.

WALSHER (IO, Link)

DATE: October 31, 1972

TITLE: 1-D WALSH TRANSFORM

TYPE: Phase

PARAMETER: IO - if IO is #1 then input is on SYS001
if IO is #2 then input is on SYS002

Link - Name of phase to Link with after 'WALSHER' is over.

USAGE: Perform 1-D WALSH TRANSFORM on REAL*4 Data.

IV. ECL VICAR LIBRARY

1. VICAR SYSTEM

The VICAR (Video Image Communication and Retrieval) system is a relatively easy method of performing image processing on a production basis. It can ideally be used by scientific research personnel with little knowledge of systems-programming.

The VICAR "language" was developed in 1968 as a joint effort of the JPL (Jet Propulsion Laboratory) and IBM. A current OS version of VICAR is now operational on the IBM 360/44 of ECL.

At JPL, VICAR has evolved over the years to contain more than a hundred application programs, which are the building blocks for a chain of processes to be performed on an image. A researcher can use a combination of these programs(by following the description of each program and the options available in each), without going into the details of each block.

A VICAR programmer, who designs, codes and debugs a program, can make use of the "VICAR Problem-Programmers Guide" and various other documentation supplied by JPL; however the programmers are advised to check with the VICAR personnel at the Image Processing Institute, about the changes peculiar to ECL's IBM 360/44 Operating System. In addition there may be additional changes in the system, such as roll-jobs and other options. Up-to-date information about ECL's VICAR System is reported as "VICAR NEWS" in "IPLNEWS" file.

In the following section is a list of application programs currently available on ECL's IBM 360/44. In addition, several application programs, which were not available from JPL have been developed at the Image Processing Institute. Examples are FOUFIL and HADFIL, which perform filtering in the 2-dimensional discrete Fourier and Hadamard transform domains, respectively.

2. LIST OF VICAR APPLICATIONS PROGRAMS AND DESCRIPTIONS

The following list gives the names of VICAR applications programs followed by a short description. They are written either in Fortran or Assembly language and differ from the conventional Frotran subroutines in their I/O structure. The user should study the description, parameters and options of each program, which is available at the Image Processing Institute. It should be noted that, some of the programs are currently not applicable at ECL, since they are designed for applications which make use of the hardware available only at JPL.

ADDON	POSITIONS A TAPE AFTER THE LAST FILE SO ADDITIONAL FILES MAY BE ADDED TO THE TAPE.
ADL	ADDS OR SUBTRACTS DN VALUES TO PIXELS THAT ARE ON A DIAGONAL LINE SPECIFIED BY ITS END POINTS. 'ADL' CAN BE USED TO SEPARATE OBJECTS THAT ARE CLOSE TOGETHER BY DRAWING A LINE BETWEEN THEM, OR IT MAY BE USED TO DRAW ARBITRARY LINES ON A PICTURE.
AFILTER	IS SIMILAR TO 'FILTER', BUT IT IS DESIGNED FOR USE WITH ASYMMETRICAL FILTERS. THE ENTIRE WEIGHT MATRIX MUST BE SPECIFIED.
APAVG	REDUCES PICTURE SIZE AND NOISE BY AVERAGING A N BY M ARRAY OF PICTURE ELEMENTS FOR EACH OUTPUT PIXEL. THE MAXIMUM NUMBER OF LINES AVERAGED IS FOUR.
ASTRTCH2	PERFORMS AN AUTOMATIC LINEAR CONTRAST STRETCH ON AN INPUT PICTURE AND CERTAIN COMBINATIONS OF THE FOLLOWING OUTPUTS ARE AVAIABLE. A STRETCHED OUTPUT PICTURE, A STRETCHED MASKED PICTURE SUITABLE FOR VFC PLAYBACK, A STRETCHED POLAROID, A PRINTER LISTING OF THE HISTOGRAM OR CUMULATIVE DISTRIBUTION FUNCTION. THE AUTOMATIC STRETCH COMPUTATION MAY BE SPECIFIED TO SATURATE A PERCENTAGE OF THE INPUT PICTURE AT 0 DN AND THE HIGH DN RANGE (255 DN OR 511 DN). ALTERNATIVELY, IT MAY BE SPECIFIED AS A NUMBER OF DIFFERENT STRETCHS ABOUT EITHER THE PEAK OR MEAN ON OF THE INPUT PICTURE.
AUTOSAR	FINDS PIXELS WHOSE DN DEVIATES BY MORE THAN A USER SPECIFIED TOLERANCE FROM THE AVERAGE OF THE CORRESPONDING PIXELS IN THE IMMEDIATELY PRECEDING AND SUCCEEDING LINES. REPLACES THOSE PIXEL DN VALUES WITH THE AVERAGE.
BOXFLT	APPLIES A LOW PASS FILTER (AVERAGING) TO AN INPUT PICTURE. THE FILTER WEIGHTS ARE SET AT UNITY OVER A USER SPECIFIED AREA OF ONE OR TWO DIMENSIONS.
CONCAT	MAKES ONE COMPOSITE PICTURE FROM UP TO TEN INPUT PICTURES EACH OF THE SAME SIZE.
CROSS	DETERMINES THE HORIZONTAL AND VERTICAL TRANSLATIONS REQUIRED TO REGISTER A RECTANGULAR AREA OF ONE PICTURE WITH A CORRESPONDING RECTANGULAR AREA OF ANOTHER, BASED ON DISPLACEMENT AT WHICH PICTURE DN'S DIFFER THE LEAST.

DIFFPIC	SUBTRACTS OR ADDS TWO PICTURES AND SCALES THE RESULT TO USER SPECIFICATIONS.
DISPLAY	DISPLAYS A PORTION OF A PICTURE ON THE LINE PRINTER. IT USES CHARACTERS TO EMPHASIS DIFFERENCES IN GRAY LEVELS.
EXPAND	ENLARGES A PICTURE N TIMES BY REPEATING EACH PIXEL IN AN N BY N ARRAY RATHER THAN BY INTERPOLATION.
FASTFIL1	PERFORMS A HIGH PASS FILTER OF THE INPUT PICTURE WITH AN ALGORITHM DESIGNED TO ELIMINATE RINGING AT THE EDGES (CAUSED BY A DARK BORDER AROUND THE PICTURE.) A ONE DIMENSIONAL (LINE BY LINE) FILTER IS USED.
FASTFIL2	A TWO DIMENSIONAL VERSION OF 'FASTFIL1' WITH A LOWPASS FILTERED PICTURE AND A HIGH PASS HISTOGRAM OPTIONALLY AVAILABLE.
FFT2	COMPUTES DIRECT AND INVERSE TWO-DIMENSIONAL, COMPLEX FOURIER TRANSFORMATIONS OF REAL IMAGE-LIKE DATA SETS. TRANSFORMATIONS MUST HAVE VERTICAL AND HORIZONTAL DIMENSIONS EQUAL TO A POWER OF TWO.
FFTPIC	PROCESSES COMPLEX FOURIER TRANSFORMS OR COMPLEX PICTURES TO EXTRACT, REORGANIZE, AND AUTOMATICALLY SCALE VARIOUS USER SPECIFIED FUNCTIONS OF THE DATA FOR IMAGE DISPLAY.
FILTER	ALTERS THE SPATIAL FREQUENCY SPECTRUM OF A PICTURE BY CONVOLVING THE PICTURE WITH AN INPUT FILTER FUNCTION. MOST OFTEN USED TO PRODUCE HIGH PASS, LOW PASS, OR MTF ENHANCED IMAGES. 'WTGEN' CAN BE USED TO GENERATE APPROPRIATE CONVOLUTION MATRICES.
FIT	SCALES A HALFWORD PICTURE (2 BYTES PER PIXEL) TO FILL THE DN RANGE OF 0 TO 255 AS A BYTE OUTPUT PICTURE.
FLOT	CAN BE USED TO ROTATE A PICTURE 90 DEGREES (CLOCKWISE OR COUNTER CLOCKWISE) OR TO REFLECT IT ABOUT EITHER THE VERTICAL OR HORIZONTAL AXIS.
FOTO	PRODUCES A HARDCOPY POLAROID PRINT OF AN INPUT PICTURE.

GEN	GENERATES A PICTURE GIVEN AN INITIAL VALUE, A HORIZONTAL INCREMENT, AND A VERTICAL INCREMENT. 'GEN' IS USED TO FORMAT DISC DATA SETS AND TO GENERATE TEST PICTURES.
GEOM	PERFORMS GEOMETRIC TRANSFORMATIONS. INPUT QUADRILATERALS ARE TRANSFORMED TO RECTANGLES. ALSO USED TO MAGNIFY OF DEMAGNIFY PORTIONS OF AN INPUT IMAGE. SHOULD BE USED WHEN ONE LINE OUTPUT IS COMPUTED FROM A SMALL NUMBER OF INPUT LINES.
GRID	USED TO OVERLAY A GRID NETWORK ON A PICTURE. PARAMETERS PROVIDE CONTROL OF THE GRID SPACING AND THE DN INTENSITY OF THE GRID LINES AND MARKS. THE DEFAULT PARAMETERS PROVIDE FOR A NETWORK WHICH IS ALIGNED WITH THE REFERENCE MARKS OF THE 'MASK' PROGRAM.
INSECT	INSERTS A RECTANGULAR PORTION OF A PICTURE INTO AN ARBITRARY LOCATION OF A SECOND PICTURE.
LAVE	COMPUTES THE AVERAGE OF A LINEAR ARRAY IN EITHER THE VERTICAL OR HORIZONTAL DIRECTIONS. THE AVERAGE IS OUTPUT AS A SINGLE LINE AND MAY BE PLOTTED WITH THE PROGRAM 'LPLOT'.
LGEOM	SAME AS 'GEOM' EXCEPT OPTIMALLY EMPLOYED WHEN ONE OUTPUT LINE IS COMPUTED FROM A LARGE NUMBER OF INPUT LINES.
LICORGEN	PRODUCES CORRECTION COEFFICIENTS TO PREFORM EITHER A FIRST ORDER PHOTOMETRIC CORRECTION RELATIVE TO A PARTICULAR PHOTOMETRIC DEVICE OR TO DETREND A PARTICULAR PICTURE. 'LICOR' CAN BE FETCHED OR LATER EXECUTED TO PREFORM THE ACTUAL CORRECTION.
LIST	PRINTS PIXEL-BY-PIXEL LISTING OF ANY PORTION(S) OF A PICTURE. 'LIST' WILL PRODUCE A HISTOGRAM OF THE PRINTED AREA(S), AND/OR LIST THE PICTURE LABELS.
LITEXFER	USED FOR GENERATING THE LIGHT-TRANSFER CHARACTERISTICS OF UP TO FIFTY DIFFERENT AREAS WITHIN AN IMAGE USING UP TO TEN INPUT FLAT FIELDS. THE OUTPUT IS IN THE FORM OF PRINTED TABLES AND/OR OPTICAL PLOT TAPE FOR THE CALCOMP PLOTTER.
LPLOT	PLOTS DN VALUES ALONG A LINE PASSING THRU THE PICTURE IN ANY DIRECTION. LINEAR INTERPOLATION

	IS USED TO PROVIDE A CONTINUOUS PLOT FOR THE CALCOMP PLOTTER.
MAG	GENERATES PARAMETERS AND FETCHES 'GEOM'. PARAMETERS CAN BE GENERATED TO MAGNIFY PICTURE SIZE, REDUCE PICTURE SIZE, ALTER ASPECT RATIO, OR SKEW A PICTURE.
MAPGRID	INSERTS A REFERENCE GRID OF ALTERNATE BLACK AND WHITE BARS INTO A PICTURE.
MASK	USED TO WRITE A PICTURE ON TAPE FOR VFC PLAYBACK, PROVIDES GRAY SCALES, REFERENCE MARKS, AND ANNOTATION. AN OPTIONAL HISTOGRAM MAY BE INCLUDED.
MOSAIC	MOSAICS UP TO TEN INPUT PICTURES INTO ONE OUTPUT PICTURE RETAINING DN'S ABOVE A SPECIFIED THRESHOLD.
PICAVE	CREATES AN AVERAGE PICTURE FROM UP TO TEN INPUT PICTURES WHICH MAY BE LINEARLY DISPLACED FROM ONE ANOTHER. 'PICAVE' CAN BE USED TO REMOVE GRAIN NOISE FROM PICTURES.
PIXH	PROVIDES FRACTIONAL FIXED POINT ARITHMETRIC OPERATIONS ON PICTURES USING HALFWORD STORAGE. OPERATIONS INCLUDED BYTE TO HALFWORD, HALFWORD TO BYTE, DIVISION BY A CONSTANT, ETC.
PIXPIK	REDUCES AN INPUT PICTURE BY EXTRACTING EVERY MTH SAMPLE OF EVERY NTH LINE FROM THE INPUT PICTURE.
POWER	COMPUTES THE ONE-DIMENSIONAL POWER SPECTRUM OF A SPECIFIED PORTION ON A PICTURE. THE SQUARE ROOT OF THE POWER SPECTRUM IS DISPLAYED ON THE LINE PRINTER AND A CALCOMP PLOTTER TAPE IS PRODUCED.
PSAR	ADJUSTS THE DN OF ALL POINTS WITHIN USER SPECIFIED POLYGONAL AREAS. BOUNDARY EXCLUSION IS OPTIONAL AND 'PSAR' CAN ALSO GENERATE PCTURES IN BOTH BYTE AND HALFWORD.
QSAR	USED TO ADD OR SUBTRACT DN VALUES TO PIXELS IN ARBITRARY RECTANGLES OF A PICTURE. PRIMARILY USED TO CORRECTED FLAWS, AND (IN CONJUNCTION WITH 'GEN') TO GENERATE TEST PICTURES.
REGISTER	REGISTERS TWO PICTURES BY CORRELATING COMMON OVERLAP BETWEEN THEM. PARAMETERS ARE GENERATED FOR AN EXECUTION OF 'GEOM'. THIS PROGRAM MAY BE RUN INTERACTIVELY.

ROTATE	USED TO ROTATE AN INPUT PICTURE 90 DEGREES CLOCKWISE OR COUNTERCLOCKWISE.
ROTATE2	ROTATES A PICTURE ANY NUMBER OF DEGREES ABOUT A SPECIFIED POINT IN THE INPUT AND PLACES THE CENTER OF ROTATION AT ANY POINT IN THE OUTPUT. 'ROTATE2' GENERATES THE NECESSARY PARAMETERS AND THEN FETCHS 'LGEOM' OR 'GEOM' TO ACCOMPLISH THE ROTATION.
SAR	USED TO COPY PICTURES FROM ONE DATA SET TO ANOTHER, AND TO REMOVE BLEMISHES AND LINES FROM A PICTURE BY FILLING IN THE RECTANGLE WITH AN AVERAGE FROM SURROUNDING PIXELS.
SCANLOG	LOGS A VFC SCAN OF A FILM TRANSPARENCY DIRECTLY ON THE COMPUTER. 'SCANLOG' PRODUCES A STANDARD, LABELLED 'VICAR' DATA SET FROM THE SCAN. AN OPTIONAL HISTOGRAM OF THE SCANNED DATA MAY BE PRINTED.
SCRIBE	CAN SCRIBE UP TO 30 RETANGLES AROUND AREAS IN A PICTURE. SIMPLE BACKGROUND PICTURES CAN BE GENERATED INSTEAD OF AN INPUT PICTURE.
STRETCH	USED TO CHANGE THE PIXEL DN VALUES OF A PICTURE BY GENERATING A TRANSFER FUNCTION ON THE DOMAIN OF INPUT DN VALUES.
SYNPIC	CREATES A COMPLEX PICTURE OF ZEROS AND ADDS USER SPECIFIED REAL AND IMAGINARY VALUES TO IT.
TAPELOG	PRODUCES A HISTOGRAM ON THE PRINTER AND A STANDARD 'VICAR' OUTPUT DATA SET FROM A VFC SCAN TAPE WITH PARITY ERRORS.
THRESHLD	USED TO LOCATE AND LIST AREAS CONTAINING POINTS ABOVE A SPECIFIED DN THRESHOLD.
UNITAVE	FROM A PICTURE IN BYTE FORMAT WITH REPETITIVE RECTANGULAR UNITS, 'UNITAVE' WILL CALCULATE THE AVERAGE AND PRODUCE A PICTURE WHERE THE AVERAGE OF THE UNIT REPLACES THE INDIVIDUAL UNIT.
VCOPY	USED TO COPY TAPES OR PORTIONS OF TAPES. LABEL FORMATS AND RECORD SIZES MAY BE CHANGED. PICTURE LABELS CAN BE LISTED WITHOUT COPYING THE TAPE.
VREWIND	REWINDS THE TAPE SPECIFIED AS THE PRIMARY INPUT DATA SET.

VTIO	CAN BE USED TO COPY FILES FROM TAPE TO DISK, DISK TO TAPE, AND FOR RANDOM TAPE FILE POSITIONING. VTIO IS USUALLY WRITTEN AS A FUNCTION UNDER 'VMAST' BUT MAY BE USED BY EXPLICIT USER REQUEST ON A 'EXEC' CARD. IN THE LATTER CASE EXTREME CAUTION SHOULD BE EXERCISED.
VUNLOAD	UNLOADS A TAPE AND (OPTIONALLY) PRINTS A MESSAGE TO THE OPERATOR.

3. EXAMPLE

Sample Walsh Transform and Filtering Example Under VICAR

```
//SYSRDR ACCESS VICAR
1*
RESERVE, 2, 256, 256, DISK4, &A
RESERVE, 1, 1024, 256, DISK3, BB
RESERVE, 1, 1024, 256, DISK5, CC
READ, , 001064, IN, 9B
E, DISPLAY, IN07, *, (1, 1, 256, 256)
E, VWALSH, (IN07, BB), CC, (1, 1, 256, 256),(PACK, REAL)
E, WALSFLTR, CC, BB, , (30, 150)
E, VWALSH, (BB, CC), &A, , (REAL, PACK)
E, DISPLAY, &A, *, (1, 1, 256, 256)
END
1*
```

The first two statements are the standard VICAR deck set-up. They allow the following cards to be compiled and executed under VICAR control.

The first RESERVE card creates two packed data sets (256 bytes per line by 256 lines). The data sets are on DISK4 and one has the name &A. The second one's name is not specified. This is the convention for telling VICAR it can use this data set for scratch. It will use the data set when it reads the tape. The next RESERVE statement creates one real data set (1024 bytes per line by 256 lines). It is allocated on DISK3 and named BB. The last RESERVE statement creates another real data set called CC on DISK5.

The READ statement causes tape 001064 to be read under 9B Format (9 track tape, 800BPI., non-labeled). The second field has no meaning, IN will be the name associated with this particular tape throughout the program.

The next five statements are execute statements. They control which programs are to be executed. The first one causes program DISPLAY to be executed. Input will be IN07 (file #7 on tape 001064). This program has no output data set so there must be an asterisk in field four. The fifth field specifies the output data set size (in this case the size of the picture on the printer). The output will start with line #1, sample #1 and continue through 256 lines of 256 samples per line.

The next statement executes program VWALSH. It takes a two-dimensional Walsh transform of input data set IN07. Data set BB is a work area for the program while the output is on CC. The output size parameters are the same size as before. The sixth field is the program option field. VWALSH requires two parameters; input type (REAL or PACK) and output type. Since output from all VICAR programs is packed unless otherwise noted, input to this program is packed. Because filtering is done in the real domain, the output must be real numbers. The next program, WALFLTR, has input on CC while output is on BB. This program does band pass filtering, the first number in field six is the low frequency cutoff while the second number is the high frequency cutoff. Note that field 5 is blank. When this occurs VICAR uses the input data set size to determine the output size. In this case data set CC was written in the previous program with parameters; S. L.= 1, S. S. =1, N. L. =256, N. S. =256. The next program take the WALSH transform again. This time input is on BB with CC as work area and &A as the output data set. In this case, however, the input is real and the output should be packed. The last executive displays the transformed and filtered picture on the printer.

The next statement is an END card. It defines the end of the program. The last statement is a "1*". It causes the program to actually execute.

V. ECL SAMPLE JOBS

FOURIER TRANSFORM FILTERING EXAMPLE

This sample program reads a picture from file one of an unlabeled tape and performs:

1) Change of format (packed to real).
2) Output of original framed picture on unlabeled tape after scaling and printing histogram.
3) Forward Fourier transform.
4) Low pass filtering (20% low frequency cutoff).
5) Reverse Fourier transform.
6) Output of filtered framed picture on unlabeled tape after scaling and printing histogram.

```
     1            COLUMN                                           77     80
     //SBR  2     JOB, EEEEEE,PIKOUT EXAMPLE                        1      5
   ┌ //SYSRUN    ACCESS  IPLABS
   │ // LABEL 256, SEQ =1,NL
 1 │ //SYS003 ALLOC &TEMP1,2314=SSLRES,256,FMT
   │ // LABEL 1024
   └ // EXEC UNPACKR
   ┌ //SYS004 ALLOC &TEMP3,2314=SSLRES,256,FMT
   │ // LABEL 1024
   │ //SYS003 ALLOC TAPE1,2400(800)='HP'
   │ //  LABEL 256,SEQ =1,NL
   │ //SYS002 ACCESS &TEMP1
   │ // EXEC PIKOUT
 2 │ 2,4,3
   │ 255.          0
   │ 255.          0.
   │ 4,8,0001,256
   └ /*
   ┌ //SYS001 ACCESS  &TEMP3
   │ //SYS002 ALLOC  &TEMP4,2314=SSLRES,129,FMT
   │ //  LABEL 2048
 3 │ //SYS004 ALLOC &TEMP5,2314=NEWSPL,129,FMT
   │ //LABEL 2048
   └ //EXEC PASSONE(PASSTWO,X)
```

The first statement after the Job Card assigns the phase library to be used. Since all the Image Processing phases are on IPLABS this card should be included in any program executing image processing phases.

Input to UNPACKR is on SYS002 which is allocated to data set TAPE1, 2400(800) = 001009 tells the system that the data set will be located on tape number 001009 mounted on the 800 bpi tape drive. The LABEL statement tells the system that each record of the tape is 256 bytes long and you want to read SEQ = 1 (file #1) on an unlabeled tape (NL). Output from UNPACKR is on SYS003, so it is allocated to temporary data set & TEMP1 (the data set will be formatted (FMT). The LABEL statement tells the system that each record will be 1024 bytes long. UNPACKR is now executed and unpacks the picture from tape and stores it on &TEMP1.

The next routine, PIKOUT, requires SYS002, SYS003 and SYS004. Input to PIKOUT is on SYS002 (data set &TEMP). SYS004 is the real picture output and must be allocated to a real data set. the ALLOC statement does this by allocating SYS004 to &TEMP3 (another temporary data set) located on SSLRESS (2314 = SSLRES) with 256 records and formatted (FMT). The LABEL statement specifies each record will be 1024 bytes long (256 real*4 numbers). SYS003 is the scaled, framedand packed output so it must be allocated to an output tape HP, as done earlier. Phase PIKOUT is now exectured. It reads 4 data cards. The first speciafies 2 to be the input real data set (SYS002), 4 to be the output real data set (SYS004) and 3 is the output packed data set (SYS003). The next two data cards specify the max and min CLIP and max and min DOMAIN respectively. The fourth card specifies the job option (4 produces scaling, framing, packing and histogram), 8 bits/pixel, experiment number 0001, and 256 lines in the picture. The /* ends the cards to be read by PIKOUT.

The next section takes the Fourier transforms and does filtering. These phases require the assignment of SYS001 to a real data set. This is done by accessing SYS001 to &TEMP3, which we created earlier and have no more use for the data stored in it. SYS002 is allocated to &TEMP4 which has 129 blocks with block length 2048. SYS004 is also allocated to a new data set 129 by 2048. The phase PASSONE is now executed. This phase ends by linking to a phase called TRANSONE. This phase then links to the phase specified in the option list following passone, PASSTWO. After PASSTWO is executed the X on the option list is encountered. This X terminates the job step and control goes on to the next statement. The forward Fourier transform is now completed with the results on SYS004 (DATA SET &TEMP5).

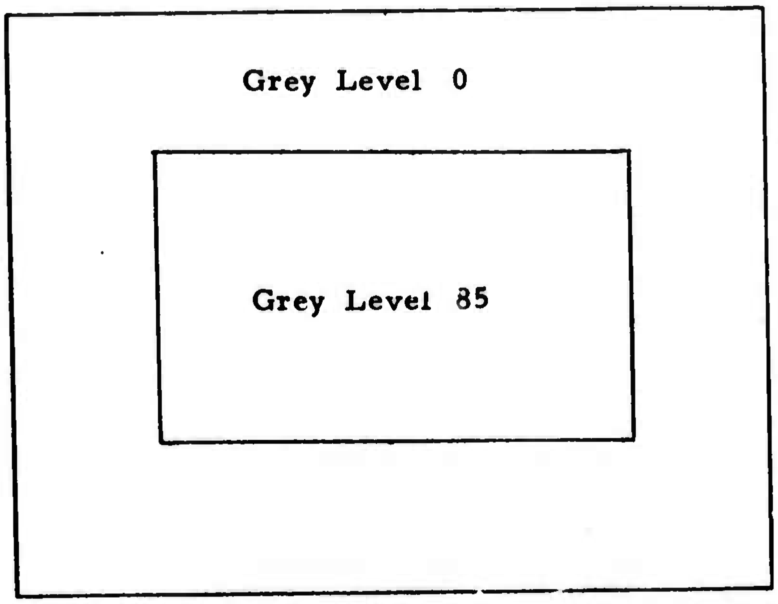
The phase FILTERS does low pass filtering, specified by the first data card. Columns 1-2 specify the number of filters to be applied (I2 FORMAT). Columns 3-4, 5-6, ... specify the filters to be used (I2 FORMAT). This program applies only one filter which is the low frequency cutoff (Option #1). The next data card specifies the low frequency cutoff (I3 FORMAT) and the high frequency cutoff (I3 FORMAT), respectively. This program passes frequencies between 25 and 128 (approximately 20% cutoff). FILTERS ends by linking to PASSTHRE which links to TRANSTWO which links to the options in the option list, PASSFOUR and then X. The reverse Fourier transform is now completed with the real picture on SYS001.

The phase PIKOUT now must be run to pack, scale, and frame the picture. SYS002 is the output real SYSUNIT and is accessed to a real data set. SYS003 is the output packed SYSUNIT and is allocated to a tape. The output will be written on SEQ = 2 (File #2) of HP. PIKOUT is now executed and the picture is scaled, frame, packed and a histogram is printed as before, except the experiment number is now 0002.

```
  // EXEC FILTERS (PASSFOUR,X)
   1 1
4  25128
  /*
  // SYS002 ACCESS &TEMP3
  //SYS003 ALLOC TAPE2,2400(800)='HP'
  // LABEL 256,SEQ=2,NL
  // EXEC PIKOUT
5 1,2,3
  255.       0.
  255.       0.
  4,8,0002,256
  /*
  1&  END OF JOB
```

TEST PICTURE CREATION EXAMPLE

Sample program creates a test image (1024 bytes by 256 lines) on data set test.



TEST IMAGE

```
//          JOB
//SYS002 ALLOC TEST,2314=PIKPAK,256,FMT,CATLG
// LABEL 1024
//MAIN44    EXEC  FORTRAN
      DIMENSION IA(256)
      DO 1 J=1,256
 1    IA(J)=0
      DO 2 I=1,256
 2    CALL DSKIO(IA,1024,I,0,2)
      DO 5 K=101,200
 5    IA(K)=85
      DO 4 I=101,200
 4    CALL DSKIO(IA,1024,I,0,2)
      STOP
      END
//     EXEC  LOADER
/&      END OF JOB
```

SAMPLE UTILITY PROGRAMS

To copy a packed picture from a 800 BPI unlabeled tape (File 1) to a disk data set (PIK1) you are going to allocate:

```
//SYS002 ALLOC TAPE,2400(800)=MYTAPE
// LABEL 256,SEQ=1,NL
//SYS003 ALLOC PIK1,2314=PIKPAK,256,FMT,CATLG
// LABEL 256
// EXEC UTILS
 COPY
```

To copy a packed picture from a 800 BPI unlabeled tape (File 2) to an already created disk data set (PIK1) and unpack it to real numbers at the same time:

```
//SYS002 ALLOC TAPE,2400(800)=MYTAPE
// LABEL 256,SEQ=2,NL
//SYS003 ACCESS PIK2
//SYSRUN ACCESS IPLABS
// EXEC UNPACKR
```

To copy a pack picture from 800 BPI unlabeled tape (File 5) to another 800 BPI unlabeled tape (File 7):

```
//SYS002 ALLOC TAPE1,2400(800)=MYTAPE
// LABEL 256,SEQ=5,NL
//SYS003 ALLOC TAPE2,2400(800)=YOURTAPE
// LABEL 256,SEQ=7,NL
// EXEC UTILS
 COPY
```

To copy a packed picture from a disk data set (PIK3) to a 800 BPI unlabeled tape (File 1):

```
//SYS002 ACCESS PIK3
//SYS003 ALLOC TAPE,2400(800)=MYTAPE
// LABEL 256,SEQ=1,NL
// EXEC UTILS
 COPY
```

To copy a REAL*4 picture from a disk data set (PIK3) to another data set (PIK4):

```
//SYS002 ACCESS PIK3
//SYS003 ALLOC PIK4,2314=ZOOPAK,256,FMT,CATLG
// LABEL 1024
// EXEC UTILS
 COPY
```

Darkened fields should be replaced with the appropriate data set names, blocksizes, file numbers, etc. for your particular job.

DISK DATA SET CREATION AND USE

To create a permanent disk data set the following statements must be used:

```
//SYS unit#  ALLOC  dsname, 2314=diskname, data  length, FMT, CATLG
//  LABEL  blocksize
```

For example:

```
//SYS001 ALLOC PIK1,2314=PIKPAK,256,FMT,CATLG
// LABEL 1024
```

These two statements allocate data set PIK1 and assign it to symbolic unit SYS001. The data set contains 256 block (Lines) and each block contains 1024 (256 real*4 records) bytes. The data is formatted and cataloged. Once a data set is created it cannot be referred to again with an ALLOC statement. If you wish to change the symbolic unit number assigned to a data set allocated previously in your job or you wish to initially assign a symbolic unit number to a data set allocated permanently in a previous job, you can use the following:

```
//SYS unit#  ACCESS  dsname
```

For example:

```
//SYS003 ACCESS PIK1
```

This statement assigns SYS003 to data set PIK1.

A temporary or scratch data set can be created and used during a job. This data set will be destroyed at the end of your job. To create a temporary data set:

```
//SYS unit#  ALLOC  &dsname, 2314=diskname, data length, FMT
//  LABEL  blocksize
```

For example:

```
    //SYSJ01 ALLOC &PIKT,2314=SSLRES,256,FMT
    // LABEL 1024
```

This created a temporary data set &PIKT which is located on disk SSLRES, has 256 blocks, each block contains 1024 bytes and is formatted. The dsname must start with & to be temporary. The ACCESS statement may be used to change the symbolic unit assignment, if necessary, of temporary data sets.

To delete a data set that YOU created and no longer want, the following statements should be used:

```
    // ACCESS dsname
    // DELETE dsname
```

For example:

```
    // ACCESS PIK1
    // DELETE PIK1
```

These two statements delete data set PIK1.

For further information on data set creation and use see the IBM 360/44 GUIDE TO SYSTEM USE.

TAPE DATA SET CREATION AND USE

Any tape used to display pictures at IPL must be 800 BPI and unlabeled. To allocate a 800 BPI unlabeled tape data set at ECL the following statements must be used:

```
//SYS# ALLOC  dsname, 2400(800)=tapename
//  LABEL  blocksize, SEQ=file#, NL
```

For example:

```
//SYS001 ALLOC TAPE1,2400(800)=MYTAPE
// LABEL 1024,SEQ=1,NL
```

This creates a data set called TAPE1 on MYTAPE which is 800 BPI and unlabeled. The data set is on file number 1 and the data set is assigned to symbolic unit SYS001.

To allocate a 1600 BPI labeled tape data set:

```
//SYS unit#  ALLOC  dsname, 2400H=tapename, CATLG
//  LABEL  blocksize, SEQ=file#
```

For example:

```
//SYS005 ALLOC INTAPE,2400H=TAPE1,CATLG
// LABEL 1024,SEQ=4
```

This statement allocates data set INTAPE on TAPE1 file number four. The blocksize is 1024 bytes and the data set is assigned symbolic unit SYS005. Now that this data set has been cataloged it can be referred to with an ACCESS statement as follows:

```
//SYS003 ACCESS INTAPE
// LABEL 1024
```

These statements assign data set INTAPE to symbolic unit SYS003. You must have a LABEL statement to redefine the blocksize although you need not redefine the file number.

To write on any tape there must be a write ring in the back of the tape. By removing this ring it protects you from accidentally writing on your tape. When you write any file on a tape, all the files following the file being written on are destroyed. Therefore, if you have data on files one through six, you cannot correct file four without destroying files five and six. The only way around this is to copy files five and six on another tape, correct file four and then copy back files five and six. Tape files should be sequential. You cannot write on file 5 until there is a file four and three and two and one.

VI. IPL OPERATIONS MANUAL

1. INTRODUCTION

The purpose of this segment is to give instruction and insight into a majority of the capabilities available in the Image Processing Laboratory. In addition to the detailed description of the options and usage of various programs, there is included a section outlining the sequence of commands involved in storing, compiling, editing, and loading user programs. Auxiliary to these instructions, section three contains a collection of information concerning messages the computer may type, solutions to the most frequent problem situations, and an index of peripheral devices with their system reference numbers.

Persons authorized to use the IPL facilities include students enrolled in the course on Image Processing and members of the staff of the Image Processing Institute or their students.

The central computer of the IPL is an HP2100A. Communication with the HP is through one of two teletype (TTY) units interfaced to it, an ASR-35 TTY (the system teletype) and a Tektronix 4010 graphics terminal. The user enters instructions as follows:

1) Sit at one of the terminals and strike any key (for example: "J"). Normally, an asterisk ("*") or the sequence "14*" will appear.

2) If this doesn't happen, ask for help. If no one is available, go to page 1-1 on "STARTING UP THE SYSTEM".

3) If step 1 works, it is possible to start any program at this point.

4) To run, for example, program TAPE, type ON, TAPE. If the 4010 CRT is being used (which is the most common situation), type ON, TAPE, 14, followed by pressing "Line Feed". The line should read "14*ON, TAPE, 14". NOTE: 14 is the logical unit number of the 4010 terminal.

5) If any program should require additional information, it will ask a series of questions. The answers to the questions may be prepared by consulting the documentation for the particular program.

2. RTE OPERATING PROCEDURES

STARTING UP THE SYSTEM

(If system has been running, but requires restart, press "Halt," go to step 7.)

1. Make sure the power is on in the lab. During the day, it should always be on, but at night or on weekends it may be initiated by pressing the START button at either end of the computer lab.

2. turn on the power switch to the I/O extender, which is on the right as you face the computer.

3. Turn on the power switch to the computer.

4. Set switch on moving head disc to 'LOAD'.

5. Turn on both teletypes. The CRT's power switch is located back under the shelf upon which the terminal is resting.

6. Ascertain that the teletype is On-Line. Now begin the loading sequense for the computer:

7. Depress button marked 'P' on computer panel.

8. Set the switches on the display register to 077760 octal.

9. Press external preset, internal preset, and "Loader Enable" button (The final button will stay lit if all is well. If it won't stay on, see following page)

10. Press RUN. Computer will halt with 102077 in display register. If this doesn"t occur, and step #9 went well, start again at step #7, after pressing "Halt".

11. Select 'S' register by depressing button 'S'

12. Press 'Clear display' button, and then RUN again. This time, the computer will run for approximately 7 seconds and then prints 'SET TIME' on the system teletype.

13. Make sure that all needed equipment is turned on: Both discs, any magnetic tape units which you intend to use (including the 3 switches under the right-most 9-track tape unit), and the Aerojet display (main power is in back).

SEQUENCE TO FOLLOW IF LOADER LIGHT DOESN'T WORK

a.) Press HALT firmly and repeatedly until the display register changes to 077761.

b.) Turn off bit #0.

c.) Press Loader Enable again (If it stays on, go to step 'd', but if not, return to 'a'.

d.) Continue.

EDITING

Line preceded by +, operator types;
Line preceded by -, computer types.

1. Store:[1]

```
+       *ON,EDIT,,,2,2
-       I/O ERR ET EQT #03              (ET=end of tape; EQT=equip-
+       *UP,3                                              ment)
+       *GO,EDIT,1
-       /EDIT: TRACKS IN NEW FILE
-       /EDIT: 02,00xx                  (xx is track # where program
-       /EDIT: END OF EDIT RUN                             is)
```

2. List from disc to:

```
      line printer
+       *LS,2,xx                        (xx is track # where program
+       *ON,EDIT,,2,6,1                                    is)

      teletype
+       *ON,EDIT,,2,1,1
```

3. Dump source to paper tape:

```
+       *LS,2,xx                        (xx is track # where program
+       *ON,EDIT,1,2,4,0                                   is)
-       /ENTER EDIT FILE
    If not editing program, type:
        /E
```

4. Edit to tape with instructions from teletype:

```
+       *LS,2,xx                        (xx is track # where program
+       *ON,EDIT,1,2,4,0                                   is)
```

Edit to tape w/instr. from tape:

```
+       *ON,EDIT,5,2,4,0
```

After both sets of commands, the computer types;

```
        /ENTER EDIT FILE
```

[1]This sequence assumes paper tape input, for punched card input, change first command to *ON, EDIT,, 13,2,2.

5. Edit to next available disc track, instructions from teletype:

```
+        *LS, 2, xx                    (xx is track # where program
+        *ON, EDIT, 1, 2, 2, 0                                 is)
```

Edit with instructions from tape:

```
+        *LS, 2, xx                    (xx is track # where program
+        *ON, EDIT, 5, 2, 2, 0                                 is)
```

6. Edit from tape to line printer:

```
+        *ON, EDIT, , , 6, 1
```

7. Compile:

```
+        *LG, 1
+        *LS, 2, xx                    (xx is track # where program
+        *ON, FTN, 2, 99                                       is)
-        $END, FTN
```

8. Assemble:

```
+        *LG, 1
+        *LS, 2, xx                    (xx is track # where program
+        *ON, ASMB, 2, 99                                      is)
-        $END, ASMB
```

HOW TO STORE ON LINE

In order to store "on line" (in other words, type in the program from the teletype), first store a one-line paper tape program, which may be found on the top of the Aerojet scanner, by using the following procedure:

```
+        *ON, EDIT, , , 2, 2 [2]
-        I/O ERR ET EQT #03                (ET=end of tape; EQT=equipment)
+        *UP, 3
+        *GO, EDIT, 1
-        /EDIT: TRACKS IN NEW FILE
-        /EDIT: 02, xx                     (xx is track # where program
-        /EDIT: END OF EDIT RUN                                      is)
```

Next, edit to next available track:

```
+        *LS, 2, xx                        (xx is track # where program
+        *ON, EDIT, 1, 2, 2, 0                                       is)
-        /ENTER EDIT FILE
+        /R, 1                             (R, 1=replace line 1)
          Here is where operator
          types in program.
```

At the end of the program type:

```
         /E
```

the computer types:

```
         /EDIT: TRACKS IN NEW FILE
         /EDIT: 02, xx                     (xx is track # where program
         /EDIT: END OF EDIT RUN                                      is)
```

[2] See footnote 1 under editing.

PROCEDURE FOR TYPICAL PROGRAM LOADING SEQUENCE

Line preceded by +, operator types;
Line preceded by -, computer types.

Put source tape into the tape reader. Type:

```
+        *ON, EDIT, , , 2, 2
-        I/O ERR ET EQT #03
+        *UP, 3
+        *GO, EDIT, 1
-        /EDIT: TRACKS IN NEW FILE
-        /EDIT: 02, xx
-        /EDIT: END OF EDIT RUN
```

If editing, follow procedure under 'Editing, #5'. Then go on.

Compiling:

If program is in Fortran, type:

```
+        *LG, 1
+        *LS, 2, xx
+        *ON, FTN, 2, 99
-        $END, FTN
```

If program is in Assemby language, type:

```
+        *LG, 1
+        *LS, 2, xx
+        *ON, ASMB, 2, 99
-        $END, ASMB
```

Loading:

```
+        *ON, LOADR, 99, , 2
-        /LOADR: "GO" WITH EDIT PARAMETERS
+        *GO, LOADR, 2, 3
-        /LOADR: name READY - LOADING COMPLETE
                                        (name=name of program
                                              loaded)
```

3. INTERFACE AND DISPLAY DOCUMENTATION

PROGRAM ASCAN

Purpose: Puts black and white pictures on the screen of the Aerojet display unit, having read them from the computer disc storage.

Usage: Program asks for a data set (disc). Type the proper integer, from 1 to 36.

Program asks for # of significant data bits (6, 8, or 10). Type the proper number. Aerojet will always display the six most significant of those bits.

Typing a zero data set terminates the program.

Possible Problems:

1. 7900A disc power not on.
2. 7900A disc not in "LOAD" mode (see the "LOAD/UNLOAD" switch on bottom unit.)
3. Aerojet not turned on.
4. +5 volt power supply to left of flying spot scanner not turned on.
5. Aerojet not in "BLACK/WHITE" mode.

PROGRAM CDSPY

Purpose: Reads three frames from disc and puts them in Aerojet memory as red, green, and blue fields of a color picture. When Aerojet is in true color mode, this color picture will appear on the screen.

Usage: Program requests 3 data sets. Type them one data set per line (Follow each response with a carriage return/line feed).

Program asks for # of significant bits of data (6, 8, or 10). Type the proper integer. Aerojet will display 6 most significant bits of each color primary.

Possible Problems:

1. Aerojet not in TRUE COLOR.

2 - 6. Numbers 1, 2, 3, 4, and 6 from documentation for program ASCAN.

PROGRAM FSCAN

Purpose: To display on the flying spot scanner a picture which is stored on the computer disc, or to digitize a slide, storing result on disc.

Usage: First question: Scan or display.

Answer: Anything other than the letters "SC" is assumed to mean "Display". (e.g. "2")

Second question: Mag-tape or disc?

Answer: Same philosophy as first answer.

<u>For Display:</u>

Third question: Picture size, data set, repetitions.

Answer: Type these three responses each on its own line (i.e. follow each integer with carriage return/line feed).

Fourth question: 6, 8, or 10 bit data?

Answer: Whichever is appropriate. 10 bits are displayed, so 6-bit data is left-shifted four places, 8-bit data shifted two places, 10-bit data is not shifted.

Fifth question: Reverse or normal?

Answer: Only 256 x 256 arrays may be displayed in "reverse" orientation (which conforms to Aerojet display format). To do this, type "RE"; to omit reversal or display different size, type anything else.

Sixth question: When pictures finish displaying, computer asks "DONE".

Answer: If done, type "YE", or anything beginning with these two letters.

If not done type anything else (e.g. "2"). This will branch you back to question three.

<u>For Scanning (digitization):</u>

Third question: Enter X, Y coordinates.

Answer: The full resolution of the display raster is 1024 x 1024; the digitization light source is a 256 x 256 square somewhere in this field.

The lower right corner of this square will start at the point specified by these two coordinates. The origin is the lower right (facing the screen) corner of the 1024 x 1024 array. The X-axis is horizontal; the Y-axis, vertical.

Type the numbers one number per line

e.g. 350 return/line feed

350 return/line feed

Fourth question: Data set, repetitions.

Answer: Type data set on which the result is to be stored, followed by return/line feed. Then type a repetition factor on the next line. This should be enough to allow time to check intensity of the square, turn off lights, turn on the PMT, and adjust the PMT response for full scale.

Fifth question: See fifth question under "display".

Sixth question: DONE?

Answer: As in display, answer "YE"--- to terminate or anything else to return to question #3.

Possible Problems:

1. +5 volt supply not turned on.
2. If scanning, A/D converter not on.

IMPORTANT: No one is allowed to use the flying spot scanner except Dr. Pratt and Mark Sanders.

PROGRAM MURDD

Purpose: Digitizes a 2.56" x 2.56" picture in color and stores the 256 x 256 x 3 element result in a triple-sized array starting in data set one of the HP 7900 disc (also referred to as "the computer disc.") This array must be unscrambled using program COLOR.

Usage: Place picture on roller of the Muirhead transmitter in such a way that the significant portion of the picture begins on the first line of the drum (left edge), immediately below the upper fastening bracket. For 'right-side-up' display on the Aerojet following disc storage, the top of the picture should follow the curvature of the left edge of the roller.

Once mounted, turn on the power to the Muirhead, press the start button, and then run MURDD.* Digitization begins when the light source passes the white band on the roller, if the program has been started by that time.

When the interrupt lights of the computer stop flashing once per second and come on solid again (after approximately 4 1/2 minutes), turn off the transmitter power and run COLOR. When this program stops, 8-bit fields of red, green, and blue will be on data sets 4, 5, and 6, respectively.

Possible Problems:

1. +5 volt supply off or A/D converters off.
2. 7900A disc not on or not in "LOAD" mode.

* instructions for running program are on the next page.

MURDD: (running instructions)

A. First answer with a "1" for scan mode.

B. How to answer parameter question which computer asks:

1) Elements to skip: 28 + N*100, where N= number of inches to skip below the upper clamp before start of sampling.

2) Line length: 256

3) Lines to skip: M*93, where M= number of inches to skip from left edge of picture.

e.g. 28, 256, 0

PROGRAM MARK

Purpose: Loading and rotation of colors into pseudo color memory.

Usage: The usage of this program may be classified into three levels of complexity: Standard, intermediate, and advanced.

Standard Operation

Type 'ON, MARK'. Turn on bit #5 on the computer. Computer types message. At this point, there are two options.

1.) Displaying a color chart. Turn off bit #6, and type a chart number from one to twenty-five (most charts of interest are from one to fourteen-- see documention on "Pseudo color charts"). This will initially load the pseudo color memory. The computer types the message "ENTER INCREMENT". Before answering, turn off bit #5. Type any small number, such as 1, or -1. The chart will rotate until bit #5 is turned on again. Each time the rotation is stopped by this action, the message "ENTER INCREMENT" will appear on the console. The program is exited by typing a zero in response to the question.

2.) Storing a color chart from paper tape. Turn on bit #6. Place the pseudo color data tape in the paper tape reader, turn on the reader, press the button marked "READ" on the reader, and type the chart number which you wish to assign to the data. If in doubt, consult the table of current pseudo color charts found in this manual. Any chart positions without a name are available. To have your chart made permanent, see Mark Sanders.

It should be noted that to store a chart, the removable disc pack in the disc memory must be write enabled. This condition will not be in effect if the current pack is a demo pack. Therefore, switch to a work pack, following the procedure for 'Switching disc packs', also found in this documentation.

The format of a data tape is as follows: Sixty-four sets of three integers, separated by commas, and followed by a "Return, Line-Feed". The integers must be between zero and fifteen, inclusive, and their sequence is interpreted as Green, Red, Blue. At present, there is n. facility for storing this data from cards.

Intermediate Operation

The value entered for the increment factor determines two things. If the value is an integer greater than or equal to one, it will be taken to mean the number of steps to rotate the chart between increments. If the value is a real fraction between -1. and 1., it will be interpreted as the inverse of the number of wait loops to execute between single-step increments.

To pause the incrementing process at a given step without being asked for a new increment, turn bit #15 on. To resume, turn it off again.

Advanced Operation

The color chart may be incremented in two, four, or eight segments, if so desired. For example, to rotate in two segments, load the chart normally, but when the computer types "ENTER INCREMENT" type, for example, '1, 2'. This would rotate the segments in steps of one. To rotate in steps of three, type '3, 2'. To rotate in the opposite direction in steps of one, type '-1, 2'. To rotate four segments in a backwards direction in steps of one, type '-1, 4', etc.

Note that the zero position in the color chart is always held constant. This is because the background area of the screen is made black in black and white or color modes by storing zeros for those positions, in the Aerojet memory. Therefore, in pseudo color, the color assigned to grey level zero will appear wherever grey level black was present in black and white. The constant assignment of this location is also in effect for other segmentation of the color chart rotation.

During the time that a color chart is rotating, in any of the possible segmentation configurations, various segments may be changed to a solid color by turning on corresponding bits of the console. The correspondence of bits to segments is: Bit #14 - Segment #2 (if present). Bit #13 - Segment #3 (if present). Bit #12 - Segment #4, etc. This continues for as many segments as were entered. The color assigned to each block of locations is black under standard operation. However, by typing a second chart number at the start of the program, the first N colors of the second chart specified will be placed into the segments of the first chart at the appropriate time, where N is the number of segments currently rotating. An example is clearly necessary.

Suppose chart #1 begins (following location zero) with the colors white, orange, dark yellow, and bright yellow. To store white in locations 16-32 of color chart #4, perform the following sequence:

Type 'ON, MARK'. Turn on bit #5, and turn off bit #6. Type the chart numbers '4, 1'. Answer the question "ENTER INCREMENT" with '-.5, 4'. Turn on bit #14, to make segment #2 solid. (Note: This will occur whether or not bit #5 is on, but if it is off, the other colors will continue to increment after segment #2 has come on solid white).

Segment #1 can not be blocked out by the above procedure.

A final option involves the occasional need to preserve the color chart in its altered state. The chart is normally restored every time a new increment is entered. To suppress this action, follow the increment number and the segment value with a third number, '2'. Example: '.5, 1, 2'. Also, it should be noted that the segment value and restoration option number stay fixed until reset. Therefore, if the above options were selected, the next response could be '.5', and would be taken to mean the same thing as '.5, 1, 2'. If then restoration was desired, the proper response would be '.5, 1, 1'.

Exit of the program is still by typing a zero in the first position of the response to the question "ENTER INCREMENT".

TAPE

DATE: 5/18/72

TITLE: Mag Tape Input/Output Program (256 x 256 images)

LANGUAGE: FORTRAN II

PARAMETERS: Tape format (Number of tape tracks, type of data, and, if integer, whether in compacted form); Read/Write option; which Files to process; which disc data to access.

SUBROUTINES: DASET, DSCIO, DSKIO

EXECUTION TIME: Printed during exectuion

USAGE: Bit #2 on the console must be set in accordance with the initial printed message concerning multiple File option. (The only advantage in reading individual files is that one less message is printed out.

Under multiple file operation, the user may transfer as many as twelve floating point arrays or 25 fixed point arrays (256 x 256) at one time. As of 7/17/72, the tape file numbers must be entered in ascending order. Disc data sets may be arranged in any order.

The description "Aerojet Format" refers to an interlaced format in which each tape record contains one line from each of three entered data sets, by placing each set of four consecutive numbers in the order: Red, Green, Blue, zero, if the three data sets contain Red, Green, and Blue primaries in that order.

DISC

DATE: 5/18/72

TITLE: Intra-disc

LANGUAGE: FORTRAN II

PARAMETERS: Branching Decision parameters, data format descriptions, and data set numbers

SUBROUTINES: DASET, DSCIO, DSKIO

EXECUTION TIME: Operations requiring longer than one minute are: Picture normalization (5 min.), Transform thresholding (2 min.), Log magnitude of a transform (11 min.).

USAGE: Bit #3 on the console controls the printout of initial branching messages. With the bit on, the computer types "Enter 2 branching param's." The proper response may be determined by consulting a table of options, below. If bit #3 is off, questions will be typed as the program runs, and decision must be made individually.

OPTION TABLE

I. DUMP/ANALYZE
- A. Dump data to printer
- B. Compute picture differences
- C. Compute mean square error
- D. Find extreme values of picture

II. CHANGE FORMAT/LOCATION
- A. Float-fixed
- B. Fixed-float
- C. Floating range shift
- D. Invert values (photographically)
- E. Data set transfer

III. PREPARE FOR DISPLAY
- A. Frame fixed array
- B. Float-fixed (same as II - A)
- C. Display transform domain

DESCRIPTION OF OPERATIONS

(Each operation is preceded by its appropriate branching parameters.)

(1, 1) Dump data to printer

Asks questions as to which line of a 256 x 256 array, either in floating point or integer format, on any disc data set, is to be printer on the line printer, then prints the line and requests whether a restart is desired or whether the user wishes to return to the main body of program "DISC. If bit #1 on the console is turned on, these questions are printed on the line printer instead of the teletype, allowing faster execution.

The 256 elements of the line are printed in a block array (e.g. Row 2 will contain elements nine through 16, etc.)

(1, 2) Compute picture differences

Upon accepting three data set numbers from the teletype, the computer forms the element-by-element difference of the arrays in the first two data sets, takes the absolute value of each difference, and stores the array of these difference elements in the third data set.

This operation may be performed on real or integer arrays, and the output data set may be the same as the input data set.

(1, 3) Compute mean square error

Input to this operation are two real arrays or two integer arrays. The square of the difference of each pair of elements is taken, these squared "errors" are summed over the whole picture, and divided by the number of element pairs considered. The computer also finds the square root of this result, and prints the mean square error and root mean square error.

(1, 4) Find extreme values of a picture

Input is a real or an integer array. Output are the maximum and minimum values in that array.

(2, 1) Float-fix

Input is a floating point array on any data set. The computer also requests an integer data set for storage of the output.

First, the extreme values in the array are found and printed.

Second, the program requests a maximum and minimum clip level and a maximum and minimum output array value.

Third, all values falling outside of the range of the clip levels are set to the nearer of the two levels, while all other values are unchanged.

Fourth, the clipped array is normalized on the basis of the four entered values, through a linear mapping which results in translating each clip level into the corresponding extreme output value, while all other samples are reduces or expanded accordingly.

Finally, the output values are truncated to integers and stored in the output data set.

(2, 2) Fixed-float
Asks for a fixed input data set and a floating output data set. Converts all integers to a real format and stores them on the output data set.

(2, 3) Floating range shift
Performs the same operations as the first four steps of the float-fix operation, except that the output is to a real array on any data set area. The normalization is a simple linear mapping, as before. Input and output data sets can be one and the same.

(2, 4) Invert values (photographically)
Integer: asks for an input data set, output data set, and number of bits. In the case of six bit data, with a maximum value of 63, this operation simply subtracts each sample from 63, so that when the picture is displayed, white areas become black, and vice versa. For other bit usage, the reference value for subtraction is assigned accordingly

Floating point: Instead of asking for "number of bits," this option requests a reference value, from which all elements in the real input array are subtracted. For earlier real or integer cases, the same data set may be used for input and output.

(2, 5) Data set transfer
Without altering format, transfers a fixed or a floating array from one disc data set to another.

(3, 1) Frame fixed array
Replaces the top ten lines of an integer array, on any data set, with a grey scale, and then replaces the bottom ten lines with either (1) a grey bar with a four-digit identification number on it, for black and white results or (2) a color chart and a four-digit I. D. number, for color pictures.

If the result is in color, the computer requests three data sets upon which to operate, one for each primary. Since the values composing the frame areas must correlate with the picture range, the number of bits used for representing a picture element must be centered (six or eight). The I. D. number is entered one digit at a time, separated by commas.

(3, 2) Float-fix
Identical to option (2, 1). Included here simply for convenience, in case the operator considers this an operation of the type "Prepare for Display."

(3, 3) Transform domain display
Performs either of three operations upon a floating point array, especially a two-dimensional forward transform of a picture, to produce a visual result to aid in the analysis.

(1) Magnitude display: Finds and prints the maximum magnitude in the array, normalizes the entire array in such a way that the maximum magnitude is set to 63.0, and all other magnitudes are sealed accordingly, and truncates these magnitudes to integer form, storing them in a user-entered output data set.

(2) Log magnitude display: Takes the log of the sum of the magnitude of each sample and one, before performing the same operation as option (1). (The printed maximum magnitude, however, is the maximum before the log operation).

(3) Threshold display: Finds and prints the maximum magnitude of the array, asks for a threshold value, sets all samples above the threshold to 63 and all samples below it to zero, and stores these results in the user-entered integer output array.

PROGRAM BTAP

Purpose: Transfers 512 x 512 arrays back and forth between 9-track tape and the moving head disc.

Usage: To write - Type 'ON,BTAP'. Computer asks for READ/WRITE decision. Type '2'. Computer asks for a data set number and a file number. If a 512 x 512 image has just been scanned on the Muirhead, it will be on data set #1. Type the two numbers on the same line, separated by a comma, and of course followed by a 'LINE-FEED'. The file number should be the next available file on the magnetic tape being used.

When the file has been copied, the computer asks "RES,EX?". Type a '1' to restart, or a '2' to exit.

Special Notes - Most operations are set up for a standard 256 x 256 picture, so a 512 x 512 image will occupy the same space as four standard images. This will not affect the ability of the tape unit to find the proper file on a magnetic tape, but it will increase the time required to process a given number of the larger files. Furthermore, program BTAP computes the proper disc data sets to use such that, for example, data set number two immediately follows number one. That is, the first line of 512 x 512 size data set number two occupies the disc space normally allocated to the first two lines of 256 x 256 size data set number 5. Not all programs make this compensation, so care should be exercised to consult documentation of any program being used to handle large arrays.

To read - Type 'ON,BTAP'. Answer the READ/WRITE question by typing a '1'. Computer asks for a data set number and a file number. Make sure that all of the disc space which is specified in this answer is actually vacant, by whatever means possible (See Special notes, above). Type the two numbers requested on the same line, separated by a comma.

When the file has been read, the computer will ask "RES,EX?". Type a '1' to restart, or a '2' to exit. On both read and write operations, the tape will not rewind until the command to exit has been given. This saves the time that would otherwise be necessary for re-positioning the tape for the next operation. HOWEVER, note that in the next operation the following file will always be considered to be <u>file 1</u>.

PROGRAM BLWUP

Purpose: Expands a 256 x 256 picture on any disc data set to fill the screen of the Aerojet display unit. Does not alter the contents of the disc in the process. Each pixel becomes a 2 x 2 block on the screen.

Usage: Type 'ON,BLWUP'. Computer asks for data set, number of bits. Respond to these questions, typing both responses on the same line, separated by a comma. If device is functional, the picture will be put on the screen at this time. The mode of display is in a pseudo-random fashion rather than the customary sequential access, which may appear confusing at first.

When the picture is complete, the computer types a standard "RES,EX?" question, which should be answered by a '1' to restart or '2' to exit.

Special Option - If the initial question is answered by including a third number, it will be interpreted as a second data set number. The picture on this second data set will then appear in the middle of the screen, surrounded by the expanded version of the picture in the first data array, as the display is being produced. For example, to show a 8-bit array from disc data set #2 surrounded by expanded 8-bit data from disc data set #1, answer the initial question by typing '1,8,2'.

To abort before the picture (or pictures) has been fully displayed, strike a key to get the attention of the computer (indicated by the computer typing an asterisk) and type 'OF,BLWUP,1'.

Possible Problems:

1. 7900A disc power not on.
2. 7900A disc not in "LOAD" mode (See the "LOAD/UNLOAD" switch on bottom unit).
3. Aerojet not fully turned on.
4. +5 volt power supply to left of flying spot scanner not turned on.
5. Aerojet not in "BLACK/WHITE" mode.
6. Wrong number of bits specified.

PROGRAM FALL

Purpose: Causes the raster of the Aerojet display unit to rise or fall (picture will appear to raise or lower on the screen) a specified number of lines.

Usage: Type 'ON,FALL'. Computer asks for number of lines to move, and in what direction. Answer with two appropriate numbers (apparent from the question) on the same line, separated by a comma. For reference, remember that the standard image contains 256 lines.

The program stops by itself upon completion.

Possible problems:

1. Aerojet unit not fully turned on.
2. +5 volt power supply to left of flying spot scanner not turned on.

PROGRAM BARS

Purpose: To enter from the keyboard data for the construction of either a bar chart or a constant-level field.

Usage: Type 'ON,BARS'. Computer asks for a data set and eight levels. Respond by typing nine integers, all on one line,separated by commas. The array constructed will be 256 x 256, on the specified data set, having eight vertical bars of size 32 x 256 each. When displayed on the Aerojet, the intensity of the bars from left to right will correspond to the order in which the eight numbers were entered.

Upon completion, the computer will ask for a decision to restart or exit. Type a '1' to restart, or a '2' to exit.

Possible Problems:

1. 7900A disc power not on.
2. 7900A disc not in "LOAD" mode. (Note: problems 1 and 2 will result in normal program completion, but with no actual data transfer to the disc memory).
3. Attempting to write on a protected track. Any time the demonstration packs are in the disc unit, they will be (or should be) protected, so that data sets 19 through 36 can not be erased. However, data sets 1 through 18 are always accessible to writing, and it is these data sets which should be most often used for experimentation or temporary use.

NEW PROGRAMS

This section will be developed in time, as new capabilities are included in the system.

4. MISCELLANEOUS NOTES

When editing a program, after edit procedure is finished, program is put into a new file and erased from the previous file.

When a program is "dumped" onto paper tape it is purged from the disc files.

On the 4010-1 teletype, it is not necessary to hit the "RETURN" key, "LINE FEED" is the only one needed.

The "LS, 2, xx" command, when editing, identifies the file where the program is. If the program has not been placed in a new file, it is not necessary to repeat the above command during one sitting at the teletype. However, if the computer types:

```
/EDIT: TRACKS IN NEW FILE
/EDIT: 02, xx
/EDIT: END OF EDIT RUN
```

Operator must retype the "LS" command with the new file number.

DISPLAYING A 512 IMAGE FROM A TAPE

Symbols used:

```
  -   computer types
  +   user types (followed by line feed)
 <--  intermediate time
  [   actions to perform
```

1) Store image on disc from tape

```
                                        [mount tape
  * ON, BTAP                            put "ON LINE"
  - 1=READ, 2=WRITE                     make sure disc on
  + 1
  - ENTER D.S., FILE #
  + 1, 1      <----------- (1 minute)
  - RES, EX?
  + 2
  - BTAP: STOP 0000
```

2) Display to aerojet from disc(B&W)

```
                                        [aerojet in B&W
                                        disc on
  * ON, AJ512                           other programs terminated
  - BARS OR PIC?
  + 2
  - ENTER 3 D.S., # BITS
  + 1, 0, 0, 8  <----------- (40 seconds)
  - RES, EXIT?
  + 2
  - AJ512: STOP 0000
```

```
3) Run pseudo-color display                        [bit #5 on
                                                   |bit #6 off, bit #15 off
  * ON, MARK                                       [aerojet in MEM. CONT.
  - BIT #6.... ENTER CHART #
  + 1       (or 4, 9(grey), 12(grey), or 17)
  - ENTER INCREMENT
                                                   [bit #5 off
  + .5
              <------------------  (runs continuously)
     to stop:
                                                   [bit #5 on
  - ENTER INCREMENT
  + 0
  - MARK: STOP 0000
```

SOME GOOD THINGS TO KNOW:

A. Changing disc packs (e.g. for demonstrations)

1. First set load/unload switch on the moving head disc to "unload" position.
2. It takes about 20 seconds for the pack to stop spinning. When it stops, there is a faint click produced.
3. Pull open the door on the disc by holding it at the top just above the smoked glass window. *
4. Exchange packs.
5. Close door and set switch to "load".
6. Disc is ready again when "drive ready" light is illuminated.

B. Numbers referring to the 7900 Moving Head disc.

1. Fixed pack is logical unit eleven (decimal). Furthermore, it is subchannel zero of equipment number nine.
2. Removable pack is also part of equipment number number nine, it is subchannel one, and logical unit twelve (decimal).
3. Data sets 1-18 are on the fixed pack.
4. Data sets 19-36 are on the removable pack.

C. Some things that will cause mysterious problems.

1. If you <u>ever</u> get an error such as:
   I/O ERR NR EQT #07

   DON'T ignore it! This particular message tells you that equipment #7 is not ready for operation. And as it says in the table below, equipment #7 is the 9-track tape unit. The most frequent error is not having the unit "on line".

* If the disc power is not on, the door won't open.

C. (cont'd.)

In any case, the system will not execute any other program as long as the program which requested I/O from the specified device is still scheduled. The proper response to this message is:

```
UP,7
```

This should be typed after attempting to remove the "not ready" condition. If all is well, the program will continue normally.

2. The other common error message is:

```
I/O ERR ET EQT #03
```

This refers to an end-of-tape condition on equipment #3, the paper tape reader. It can occur naturally when a program source has been read and reached the end, or it can happen any time 10 consecutive feed frames on a paper tape are sensed. For this reason, always position tapes as close as possible to the start of the actual data. Otherwise, the operation which reads only blank frames may cause an extraneous element to be stored. The proper response is:

```
UP,3
```

If the message followed normal input of a source file, it is also necessary to type:

```
GO,EDIT,1
```

D. Aborting programs

1. The command to abort program TAPE, for instance, is:

```
OF,TAPE,1
```

2. This command is only accepted after an asterisk has appeared.

GETTING OUT OF COMPUTER PROGRAMS

This section describes what to do to return to the executive portion of the operating system, which is the starting point of all basic commands. For each program, procedures are outlined which should cover both normal exit and problem exit.

TAPE:

1. Normal exit - When the current operation is completed, computer types "1=COMPLETE RESTART, 2=RESTART-SAME FORMAT, 3=EXIT". Type '3'.

2. Abnormal - If the tape is already running, hit a key to obtain a "*" on the keyboard (Denotes attention of computer). Then type 'OF,TAPE,1'. Computer will respond "TAPE ABORTED".

NOTE: Rewind tape unit before doing anything else.

3. Exit after typing wrong response - Turn tape unit Off Line, continue answering questions. Computer will find tape unit not ready, and type "I/O ERR NR EQT #07". At this point, striking a key will produce an "*". As above, typing 'OF,TAPE,1' will abort the program. Place unit back on line and type 'UP,7' to re-enable the unit. Proceed as desired.

DISC:

1. Normal exit - Wait for option message to restart or exit, and type appropriate integer.

2. Abnormal exit - If operation in progress, strike a key to receive "*", then type 'OF,DISC,1'. Computer types "DISC ABORTED".

3. Abnormal, wrong response to last question - If initiation of incorrect operation will not corrupt your data, continue until computer begins operation, then proceed as in (2) above. Otherwise, halt computer and execute procedure for starting up the computer (Hereafter termed "re-initialization").

ASCAN:

1. Normal exit - Keep running program until asked for "WHICH DATA SET". Respond by typing '0' (zero).

2. Abnormal - Strike key while picture is being displayed. If computer gives an "*", type 'OF,ASCAN,1'. If it doesn't, re-initialization may be necessary. Repeated difficulty with displaying may indicate hardware problem. See documentation for ASCAN.